Experience the Unseen
(Color Clarity Simulation System)

Object Oriented Programming Project

Open Ended Lab / Complex Engineering Activity

Software Engineering
First Year | Second Semester (Spring Semester)
Miss Asma and Miss Sheerina

By

Sana Jamal	SE -23002
Gul Fatima	SE – 23007
Umaima	SE - 23031

ABSTRACT

This project presents a comprehensive desktop application designed to assist individuals with color blindness, offering a suite of functionalities to improve their interaction with the visual world. The application encompasses four main features:

1. **Simulator**: Utilizing advanced image processing techniques, this feature simulates the visual perception of color-blind individuals, allowing users to understand how different types of color blindness affect the perception of the environment.

2. **Corrector**: By applying specialized color filters, this tool enhances the ability of color-blind users to distinguish between colors more effectively, thereby improving their visual experience and aiding in tasks that require color differentiation.

3. **Color Detector**: This functionality identifies and displays the name of the color at the mouse pointer's location in real-time, providing immediate feedback and aiding users in identifying colors accurately.

4. **Tester**: Based on the widely recognized Ishihara color test, this feature allows users to determine whether they are affected by color blindness or not. By presenting a series of test images, it assesses the user's color perception and provides diagnosis.

The integration of these features into a single application aims to offer a valuable toolset for color-blind individuals, enhancing their visual comprehension and facilitating better interaction with their surroundings. The application leverages cutting-edge image processing and user interface design to provide an intuitive and effective solution for the challenges faced by those with color vision deficiencies.

TABLE OF CONTENTS

Table of Contents

1. LIST OF FIGURES AND TABLES

2. Technologies and Libraries Used

1. .NET Framework:
- The primary framework for building the Windows Forms application.

2. C#:
- The programming language used for writing the application logic.

3. Windows Forms (WinForms):
- **A GUI class library for creating desktop applications.**

4. OpenCV:
- An open-source computer vision and machine learning software library.
- Specifically, the **OpenCvSharp** wrapper is used to access OpenCV functions in C#.

5. OpenFileDialog:
- A class provided by Windows Forms for opening file dialogs.

6. PictureBox:
- A Windows Forms control used to display images.

7. Mat:
- A matrix data structure used by OpenCV to store images.

8. Bitmap Converter:
- A utility from **OpenCvSharp.Extensions** for converting between OpenCV `Mat` objects and .NET `Bitmap` objects.

9. Vec3f and Vec3b:
- Data structures from OpenCV representing vectors of floats and bytes, respectively, used for pixel manipulation**.**

10. VideoCapture:
- An OpenCV class for capturing video from files.

11. System.IO.Path:
- A .NET class for manipulating file and directory paths.

12. MessageBox:
- A class in Windows Forms used to display message boxes.

<u>Notable Functions and Methods</u>

1. Cv2.ImRead:
- Reads an image from a file.

2. Cv2.AddWeighted:
- Blends two images by adding them with specific weights.

3. Mat.At and **Mat.Set:
- Access and modify pixel values in a `Mat` object.

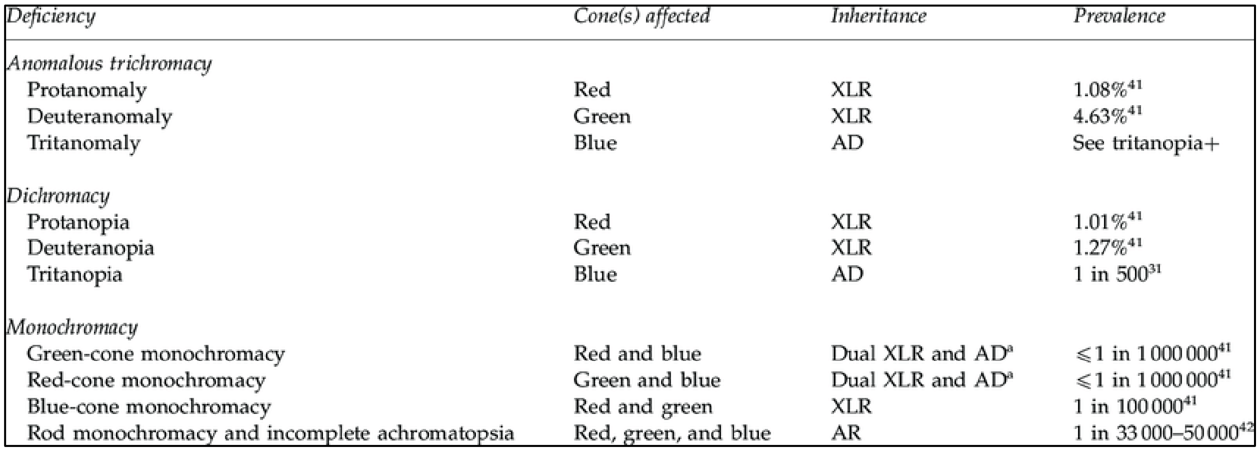
3. INTRODUCTION

3.1 Problem Statement:

Color blindness, also known as color vision deficiency, is a condition that impairs an individual's ability to perceive colors accurately. It affects approximately 8% of men and 0.5% of women globally[1], translating to millions of people who face daily challenges due to their inability to distinguish certain colors. The most common types of color blindness are red-green and blue-yellow deficiencies, with varying degrees of severity. This condition can significantly impact daily activities, educational experiences, and professional opportunities.

CVD encompasses anomalous trichromatism, dichromatism, and monochromatism, each affecting the cone receptors of the eye differently. Anomalous trichromatism, such as protanomaly and deuteranomaly, alters cone sensitivity, leading to impaired color perception. Dichromatism, including protanopia, deuteranopia, and tritanopia, results from the absence of one cone type. Monochromatism, the most severe form, causes complete color blindness due to the absence of all cone receptors.

Prevalence rates of CVD vary globally, with dichromacies affecting approximately 1% of males and a smaller percentage of females. Anomalous trichromacies and monochromatism occur at different rates across genders and ethnicities . Beyond genetic causes, CVD can also result from damage to the retina, optic nerve, or higher brain areas.



Deficiency	*Cone(s) affected*	*Inheritance*	*Prevalence*
Anomalous trichromacy			
Protanomaly	Red	XLR	1.08%[41]
Deuteranomaly	Green	XLR	4.63%[41]
Tritanomaly	Blue	AD	See tritanopia+
Dichromacy			
Protanopia	Red	XLR	1.01%[41]
Deuteranopia	Green	XLR	1.27%[41]
Tritanopia	Blue	AD	1 in 500[31]
Monochromacy			
Green-cone monochromacy	Red and blue	Dual XLR and AD[a]	⩽1 in 1 000 000[41]
Red-cone monochromacy	Green and blue	Dual XLR and AD[a]	⩽1 in 1 000 000[41]
Blue-cone monochromacy	Red and green	XLR	1 in 100 000[41]
Rod monochromacy and incomplete achromatopsia	Red, green, and blue	AR	1 in 33 000–50 000[42]

Table.01 *A summary of the different forms of congenital colour vision deficiency*

3.2 Proposed Solution:

Advancements in image processing and artificial intelligence offer promising avenues to develop effective tools for managing CVD. These technologies can simulate and correct color vision deficiencies, enhancing accessibility and usability in digital environments.

Universities play a pivotal role in students' holistic development, offering diverse extracurricular activities essential for social skills, confidence, and overall well-being. However, participation for color-blind students can be limited without adequate support.

Our desktop application aims to address these challenges by integrating a Simulator, a Corrector, Color Label tool, and a Tester using the Ishihara color test for diagnosis. This comprehensive tool aims to improve daily functionality and enhance participation in various activities for color-blind individuals.

By bridging the gap in current solutions, our application strives to empower color-blind individuals, fostering inclusivity and improving their quality of life. Integrating these tools into educational settings ensures equitable access to opportunities for all students, regardless of their color vision abilities.

3.3 Objectives

The objectives of this project are as follows:

1. Develop a Desktop Application
2. Provide a Comprehensive Suite of Tools
3. Enhance Daily Lives of Color-Blind Users
4. Foster Inclusivity, Independence, and Confidence

3.4 Scope of study

1. Target Audience:
   - Color-blind individuals of varying types and severity levels.
2. Functional Scope
   - Development of a desktop application.
   - Implementation of a visual simulator, color corrector, real-time color identification, and diagnostic tester.
3. Technical Scope
   - Compatibility with common desktop operating systems (Windows).
   - User friendly interface.

4. DESCRIPTION

4.1 Main Features

4.1.1 Methodology

i. Color blindness Simulator:

In this section, we describe the methodology used to simulate various types of color blindness, focusing on the transformation matrices applied to the RGB color values of an image. The numerical values in these transformations are derived from empirical research on how different types of color blindness affect color perception.

Transformation Matrices

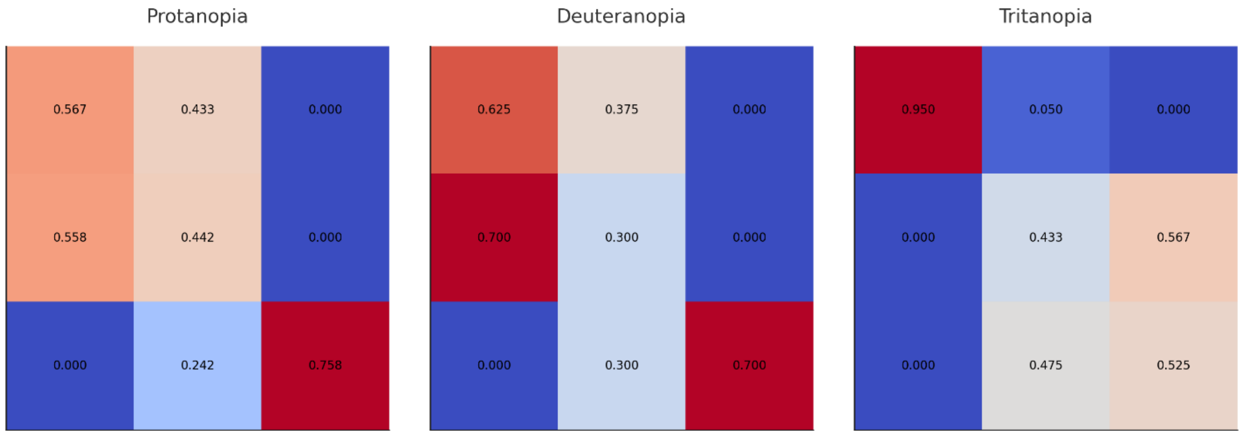
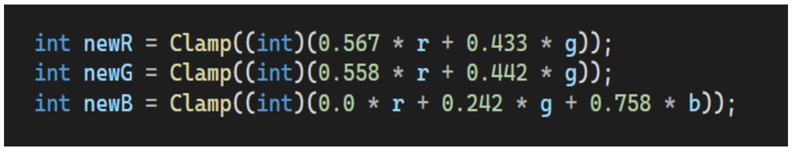


Figure.01 *Visualization of transformation matrices for Protanopia, Deuteranopia, and Tritanopia.*

1. Protanopia Simulation
Protanopia is characterized by the absence of red cone cells. The transformation matrix for Protanopia modifies the red, green, and blue color channels as follows:
Protanopia Matrix:

$$\begin{bmatrix} 0.567 & 0.433 & 0 \\ 0.588 & 0.442 & 0 \\ 0 & 0.242 & 0.758 \end{bmatrix}$$

we implemented it in our code like this:



```
int newR = Clamp((int)(0.567 * r + 0.433 * g));
int newG = Clamp((int)(0.558 * r + 0.442 * g));
int newB = Clamp((int)(0.0 * r + 0.242 * g + 0.758 * b));
```

These coefficients adjust the red and green channels to simulate the diminished perception of red, while the blue channel is adjusted to maintain overall luminance.

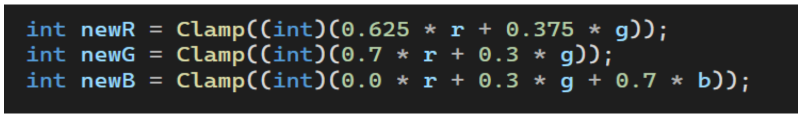
2. Deuteranopia Simulation
Deuteranopia is characterized by the absence of green cone cells. The transformation matrix for

Deuteranopia adjusts the RGB values as follows:

Deuteranopia Matrix:

$$\begin{bmatrix} 0.625 & 0.375 & 0 \\ 0.7 & 0.3 & 0 \\ 0 & 0.3 & 0.7 \end{bmatrix}$$

We implemented it in our code like this:



```
int newR = Clamp((int)(0.625 * r + 0.375 * g));
int newG = Clamp((int)(0.7 * r + 0.3 * g));
int newB = Clamp((int)(0.0 * r + 0.3 * g + 0.7 * b));
```

These coefficients modify the red and green channels to simulate the absence of green perception, while the blue channel adjustment helps maintain visual balance.

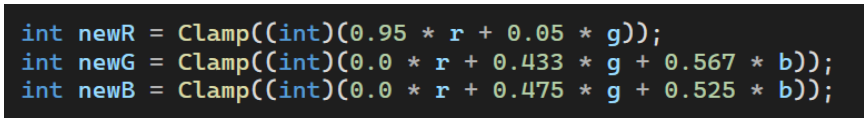
3. Tritanopia Simulation

Tritanopia is characterized by the absence of blue cone cells. The transformation matrix for Tritanopia adjusts the RGB values as follows:

Tritanopia Matrix:

$$\begin{bmatrix} 0.95 & 0.05 & 0 \\ 0 & 0.433 & 0.567 \\ 0 & 0.475 & 0.525 \end{bmatrix}$$

We implemented it in our code like this:



```
int newR = Clamp((int)(0.95 * r + 0.05 * g));
int newG = Clamp((int)(0.0 * r + 0.433 * g + 0.567 * b));
int newB = Clamp((int)(0.0 * r + 0.475 * g + 0.525 * b));
```

These coefficients simulate the diminished perception of blue while adjusting the red and green channels to maintain color balance.

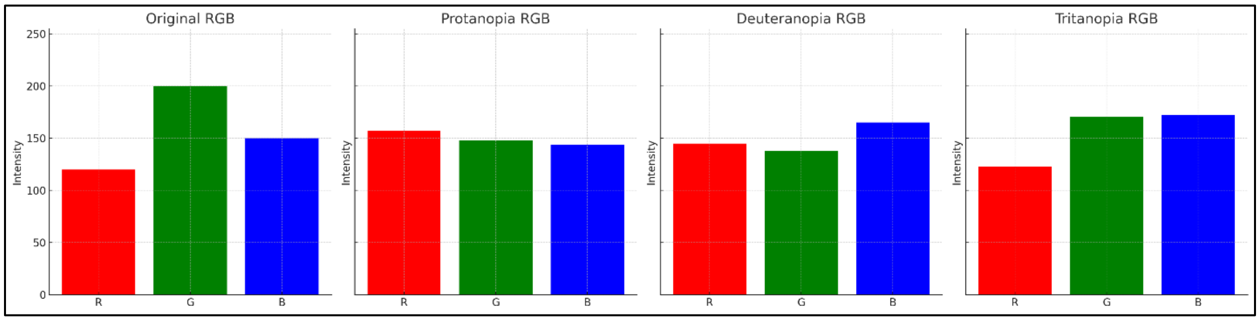


Figure.02 Comparison of RGB values before and after color blindness simulation.

ii. Color Detector

In this section, we describe the methodology used to assist individuals with color blindness by identifying and naming colors under the mouse cursor or within loaded images. The system operates within a Windows Forms environment, using RGB values and Euclidean distance calculations to provide accurate color recognition.

<u>Color Identification Process</u>

1. **Real-time Color Detection**
   - The system continuously monitors the color at the mouse cursor or within loaded images.
   - RGB values of the detected color are obtained and compared to a predefined set of colors.

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

p, q	= two points in Euclidean n-space
q_i, p_i	= Euclidean vectors, starting from the origin of the space (initial point)
n	= n-space

2. **Color Name Matching**

   - A predefined set of colors with recognizable names (e.g., "Red", "Green") is used.
   - Euclidean distance calculations determine the closest color match, ensuring accurate recognition.
   - The corresponding color name is displayed to the user.

iii. Color blindness Corrector:

Scientific Basis for Color Space Transformation

The human visual system perceives colors through three types of cone cells in the retina, which are sensitive to long (L), medium (M), and short (S) wavelengths of light. The LMS color space is a model that represents colors based on the response of these cone cells. Converting images from the commonly used BGR (Blue, Green, Red) color space to the LMS color space allows for more precise manipulation of color components to compensate for color blindness.

Conversion Matrices

BGR to LMS Conversion Matrix

The BGR to LMS conversion matrix is derived from the physiological responses of the L, M, and S cones to different wavelengths of light. The matrix is:

$$\begin{bmatrix} 0.381 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix}$$

This matrix is used to transform the BGR values into LMS values, capturing the perceptual color information as sensed by the human eye.

LMS to BGR Conversion Matrix

The LMS to BGR conversion matrix is used to transform the LMS values back into the BGR color space after enhancement. The matrix is:

$$\begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0479 & -0.2439 & 1.2045 \end{bmatrix}$$

This matrix reverses the transformation, converting the perceptually enhanced LMS values back into the displayable BGR values.

iv. Color Blindness Test

The feature designed to detect potential color blindness in users through a series of interactive visual tests. The application presents a sequence of images, each containing a number embedded within a pattern. These images are part of a standardized color vision test, such as the Ishihara test, commonly used to diagnose color blindness. Users are required to input the number they perceive in each image.
The application facilitates color blindness detection by comparing user-submitted answers with predefined correct responses to calculate accuracy as a percentage. It provides an initial assessment for potential color blindness, emphasizing intuitive usability, clear guidance, and prompt feedback. Users can navigate seamlessly through questions, submit answers, and restart the test for thorough evaluation. Designed to be user-friendly and informative, this project serves as an accessible tool for preliminary color vision screening, suitable for educational use and as an initial screening step before consulting medical professionals.

Test Setup and Implementation

- **Ishihara Color Blindness Test**: The test is based on the Ishihara color blindness plates, a widely used method to diagnose red-green color deficiencies. Each plate contains a circle of dots appearing randomized in color and size but forming a number or shape that is visible to those with normal color vision and invisible or difficult to see for those with color vision deficiency.
- **Form Initialization**: The form initializes with default values, including the total number of questions (5 in this case) and the correct answers for each question.
- **UI Components**: Utilize labels to display the question number, a picture box to show the Ishihara plates, and a text box for user input.
- **Diagnosis**:
  - **No Color Blindness**: If all answers are correct.
  - **Color Blindness :** If answers are wrong.

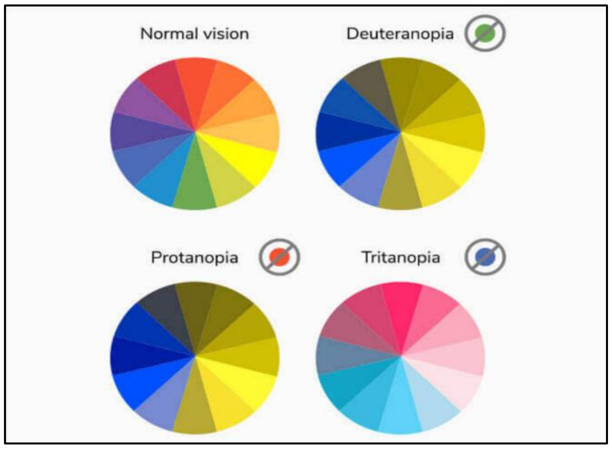


Figure.03 Piecharts describing various types of CVD

5. GUI DEVELOPMENT

5.1 SIGN IN

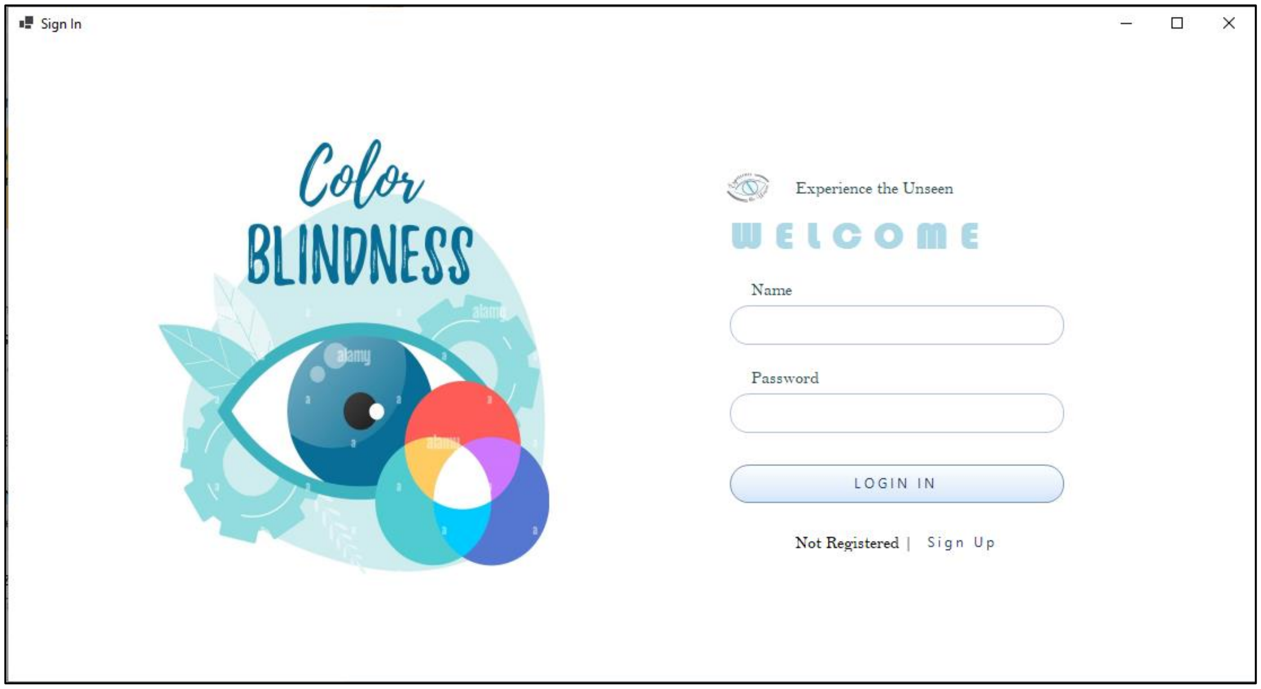
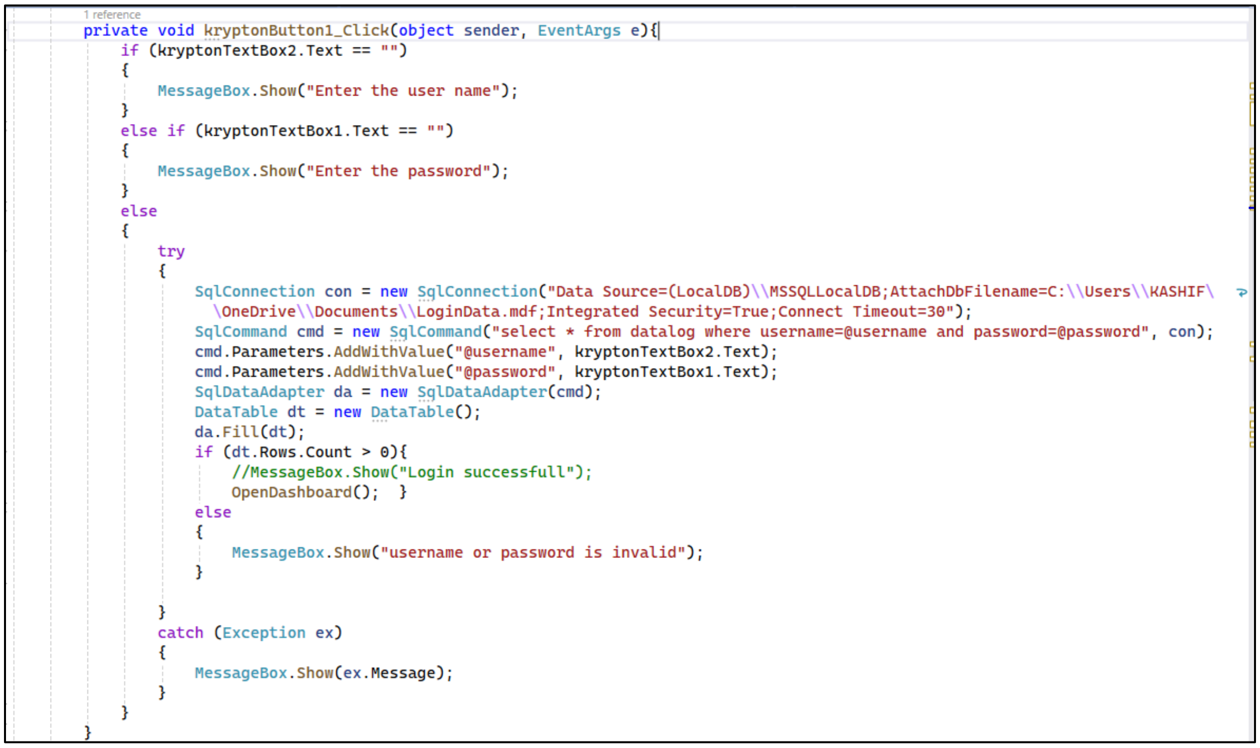


Figure.04

Code:



```
1 reference
private void kryptonButton1_Click(object sender, EventArgs e){
    if (kryptonTextBox2.Text == "")
    {
        MessageBox.Show("Enter the user name");
    }
    else if (kryptonTextBox1.Text == "")
    {
        MessageBox.Show("Enter the password");
    }
    else
    {
        try
        {
            SqlConnection con = new SqlConnection("Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\KASHIF\
              \OneDrive\\Documents\\LoginData.mdf;Integrated Security=True;Connect Timeout=30");
            SqlCommand cmd = new SqlCommand("select * from datalog where username=@username and password=@password", con);
            cmd.Parameters.AddWithValue("@username", kryptonTextBox2.Text);
            cmd.Parameters.AddWithValue("@password", kryptonTextBox1.Text);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
            if (dt.Rows.Count > 0){
                //MessageBox.Show("Login successfull");
                OpenDashboard();   }
            else
            {
                MessageBox.Show("username or password is invalid");
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

5.2 SIGN UP

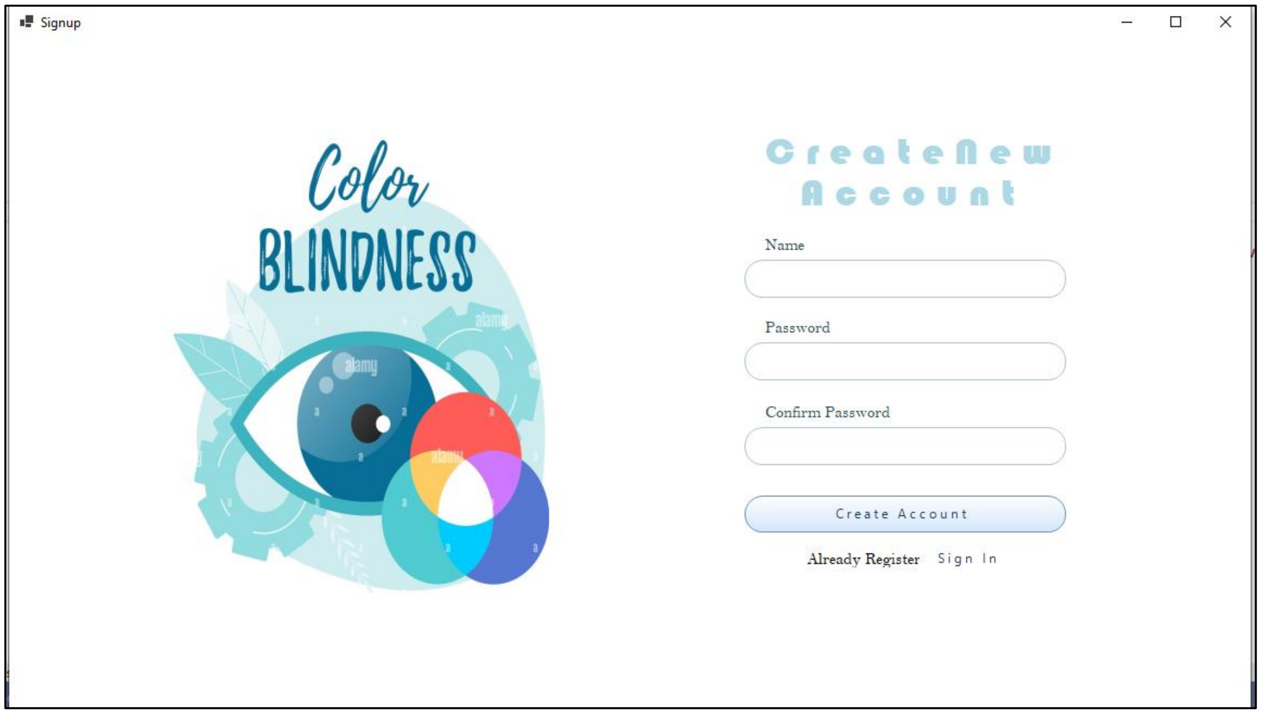
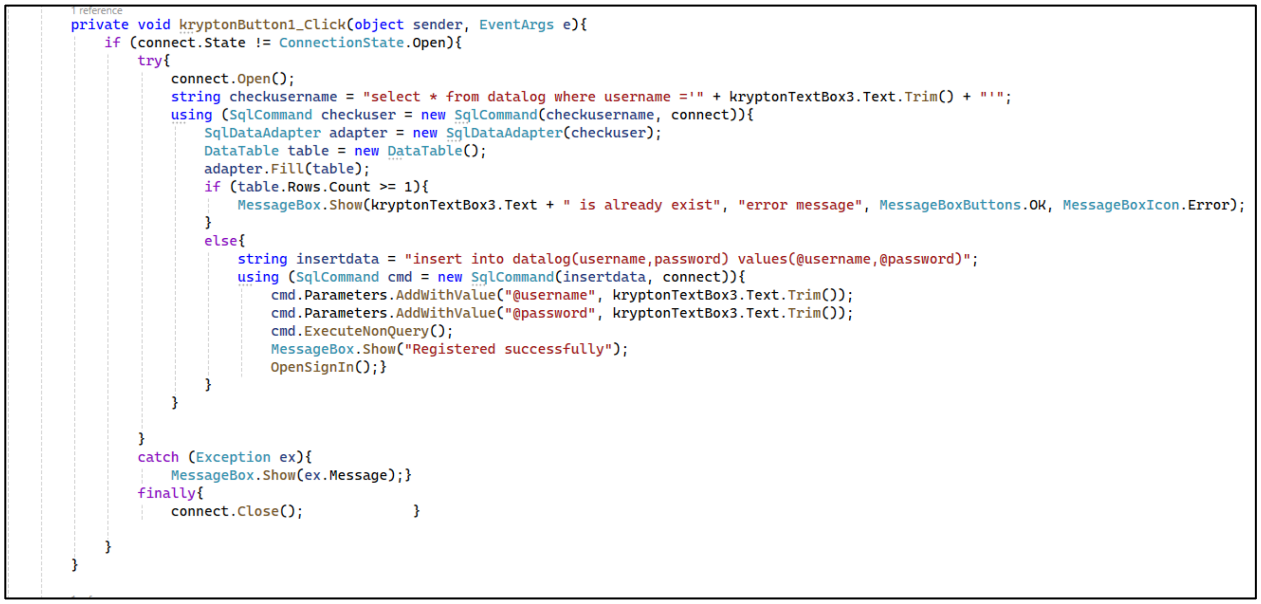


Figure.05

Code:



```
1 reference
private void kryptonButton1_Click(object sender, EventArgs e){
    if (connect.State != ConnectionState.Open){
        try{
            connect.Open();
            string checkusername = "select * from datalog where username ='" + kryptonTextBox3.Text.Trim() + "'";
            using (SqlCommand checkuser = new SqlCommand(checkusername, connect)){
                SqlDataAdapter adapter = new SqlDataAdapter(checkuser);
                DataTable table = new DataTable();
                adapter.Fill(table);
                if (table.Rows.Count >= 1){
                    MessageBox.Show(kryptonTextBox3.Text + " is already exist", "error message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                else{
                    string insertdata = "insert into datalog(username,password) values(@username,@password)";
                    using (SqlCommand cmd = new SqlCommand(insertdata, connect)){
                        cmd.Parameters.AddWithValue("@username", kryptonTextBox3.Text.Trim());
                        cmd.Parameters.AddWithValue("@password", kryptonTextBox3.Text.Trim());
                        cmd.ExecuteNonQuery();
                        MessageBox.Show("Registered successfully");
                        OpenSignIn();}
                }
            }

        }
        catch (Exception ex){
            MessageBox.Show(ex.Message);}
        finally{
            connect.Close();             }

    }
}
```

5.3 DASHBOARD

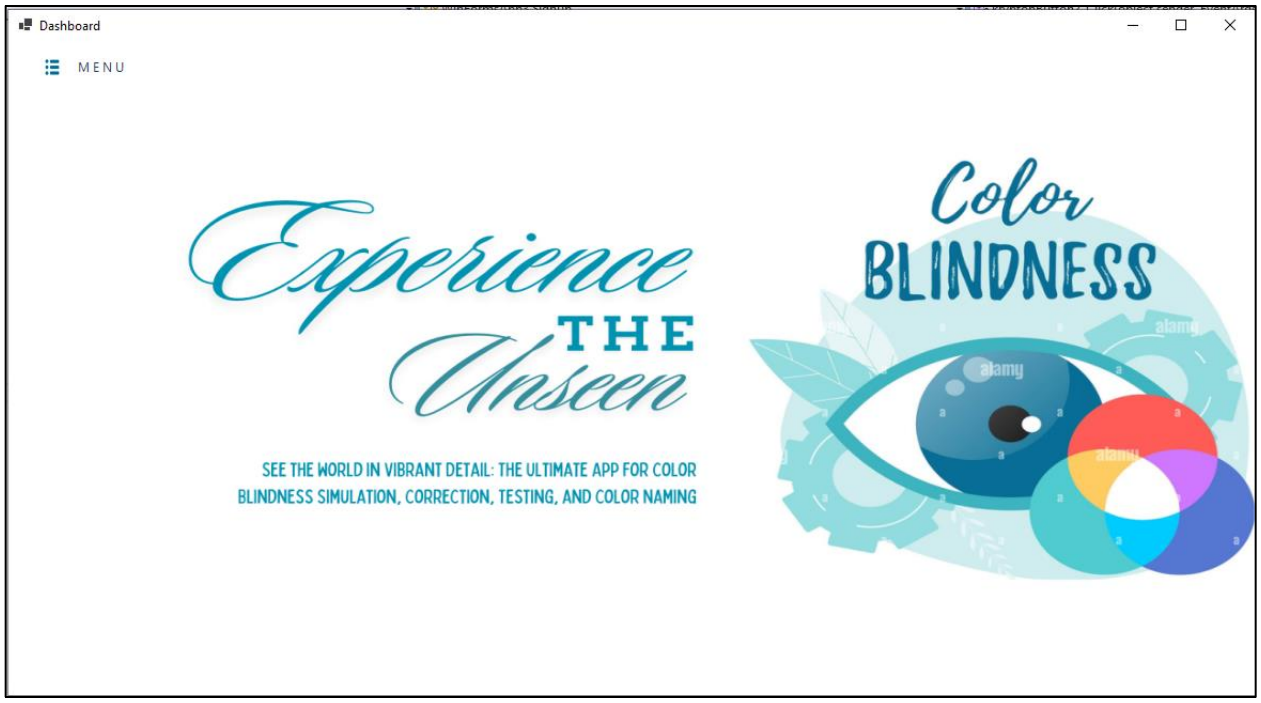

Dashboard
MENU
Experience THE Unseen
SEE THE WORLD IN VIBRANT DETAIL: THE ULTIMATE APP FOR COLOR BLINDNESS SIMULATION, CORRECTION, TESTING, AND COLOR NAMING
Color BLINDNESS

Figure.06

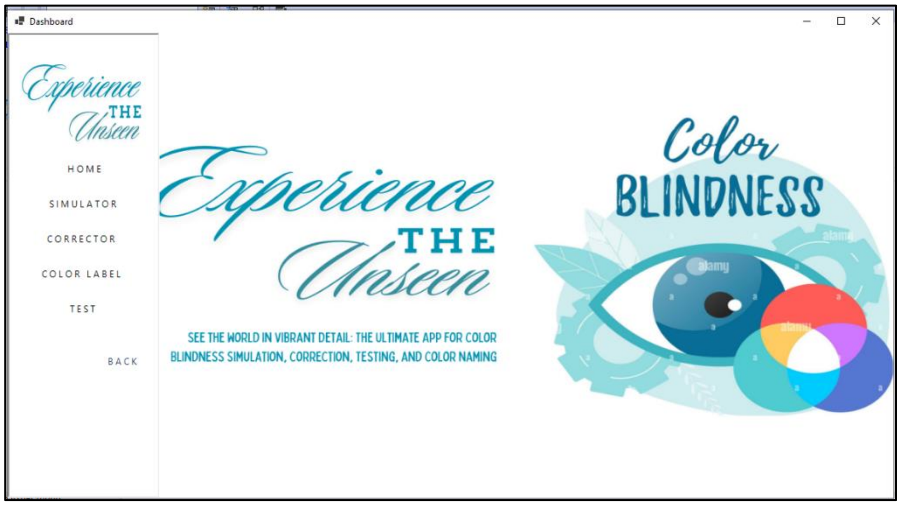
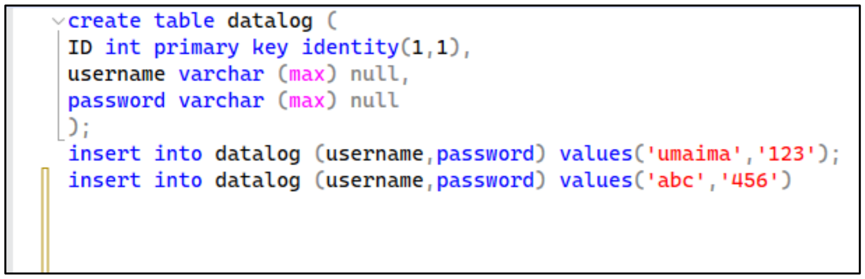

Dashboard
Experience THE Unseen
HOME
SIMULATOR
CORRECTOR
COLOR LABEL
TEST
BACK
Experience THE Unseen
SEE THE WORLD IN VIBRANT DETAIL: THE ULTIMATE APP FOR COLOR BLINDNESS SIMULATION, CORRECTION, TESTING, AND COLOR NAMING
Color BLINDNESS

Figure.07

5.4 SQL Query



```
create table datalog (
ID int primary key identity(1,1),
username varchar (max) null,
password varchar (max) null
);
insert into datalog (username,password) values('umaima','123');
insert into datalog (username,password) values('abc','456')
```

Figure.08

5.5 SIMULATOR

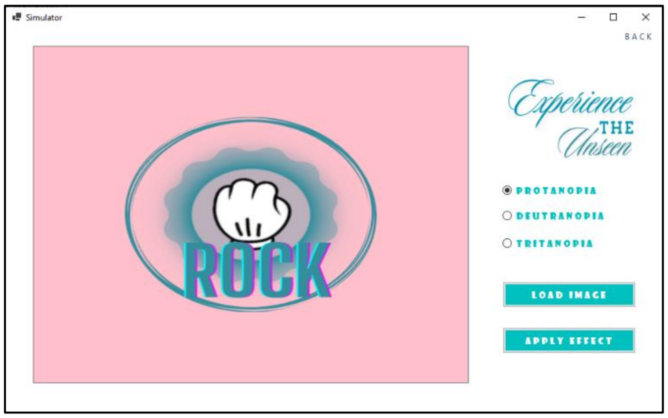


Figure.09

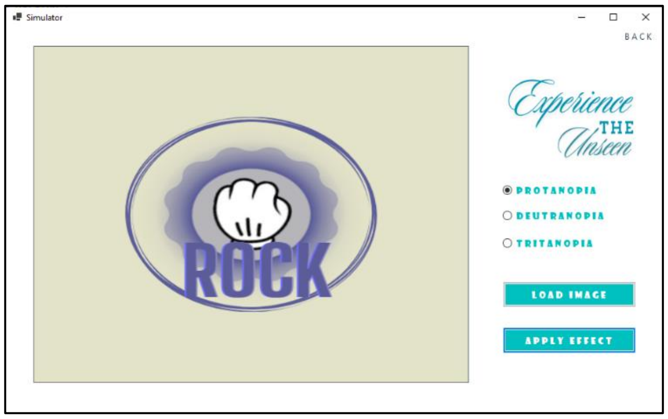
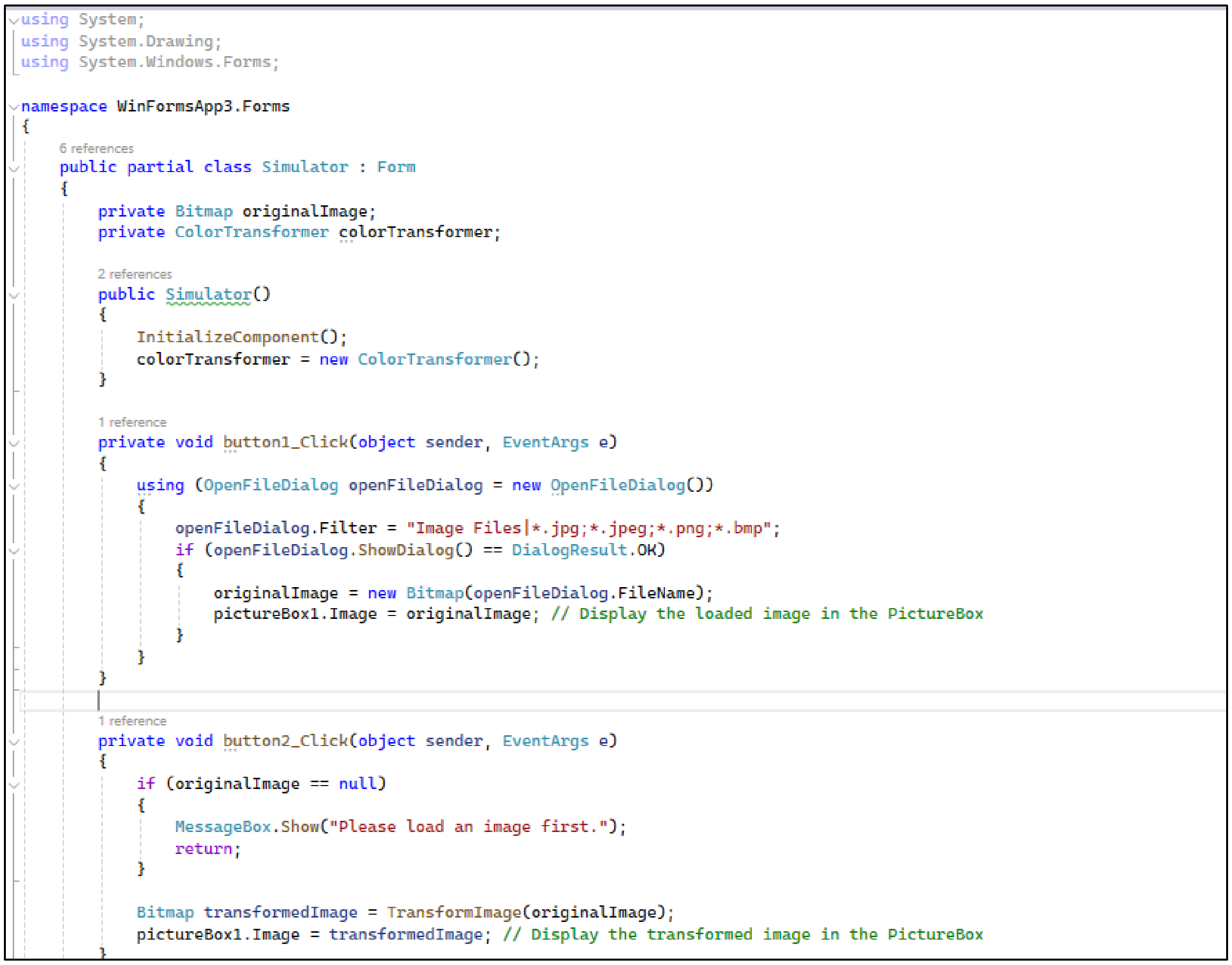
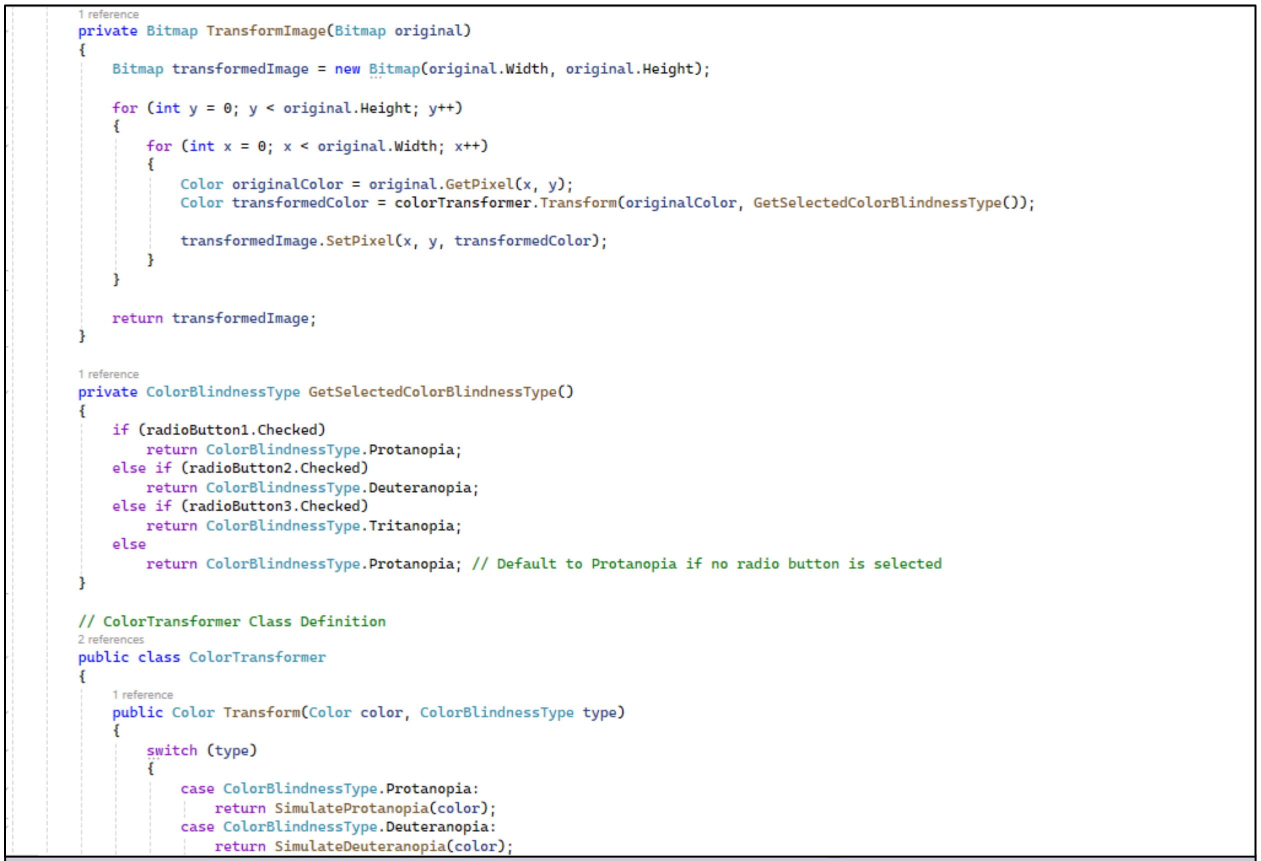


Figure.10

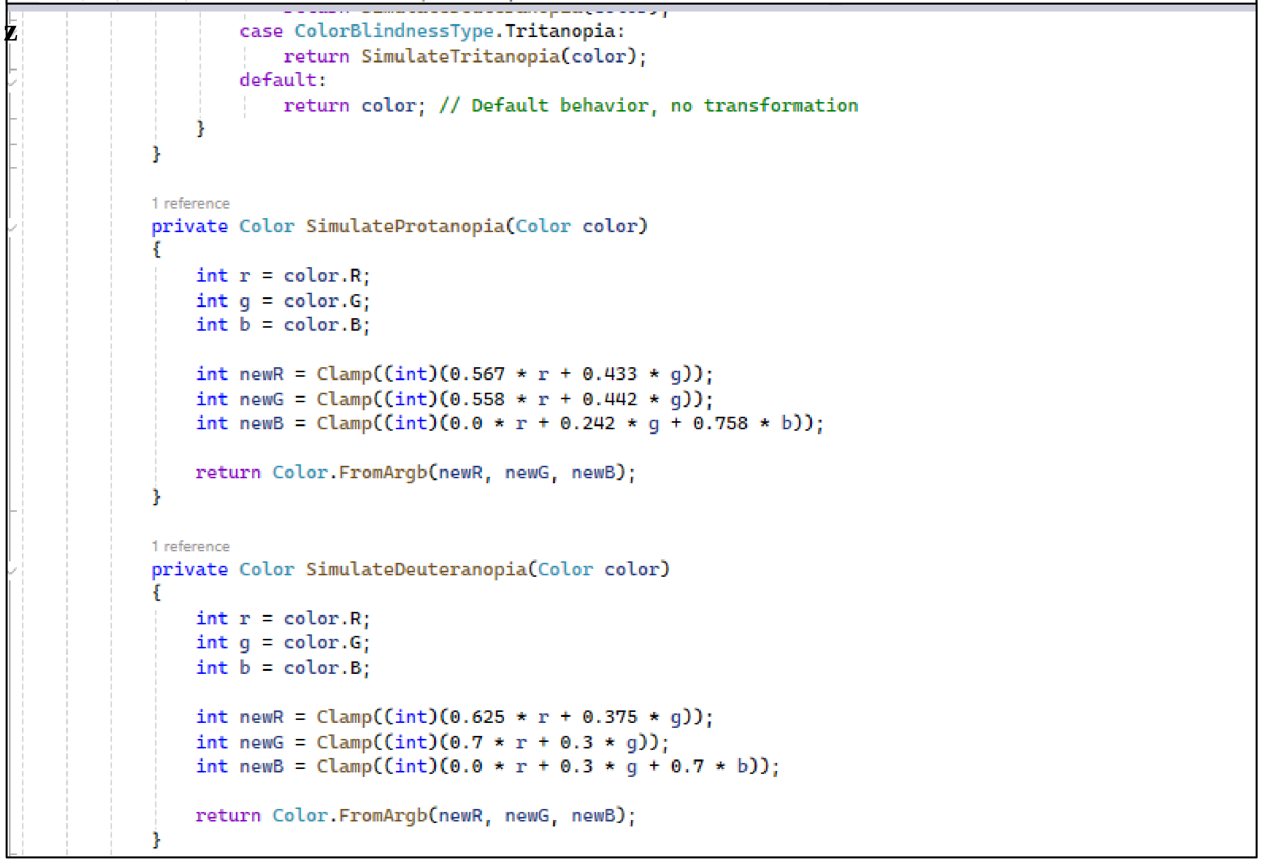
Code:



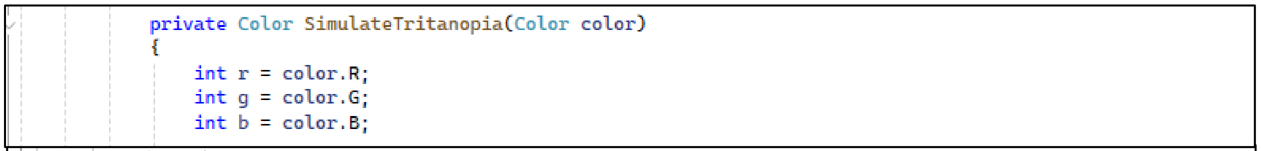
```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WinFormsApp3.Forms
{
    6 references
    public partial class Simulator : Form
    {
        private Bitmap originalImage;
        private ColorTransformer colorTransformer;

        2 references
        public Simulator()
        {
            InitializeComponent();
            colorTransformer = new ColorTransformer();
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog())
            {
                openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp";
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    originalImage = new Bitmap(openFileDialog.FileName);
                    pictureBox1.Image = originalImage; // Display the loaded image in the PictureBox
                }
            }
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            if (originalImage == null)
            {
                MessageBox.Show("Please load an image first.");
                return;
            }

            Bitmap transformedImage = TransformImage(originalImage);
            pictureBox1.Image = transformedImage; // Display the transformed image in the PictureBox
        }
```



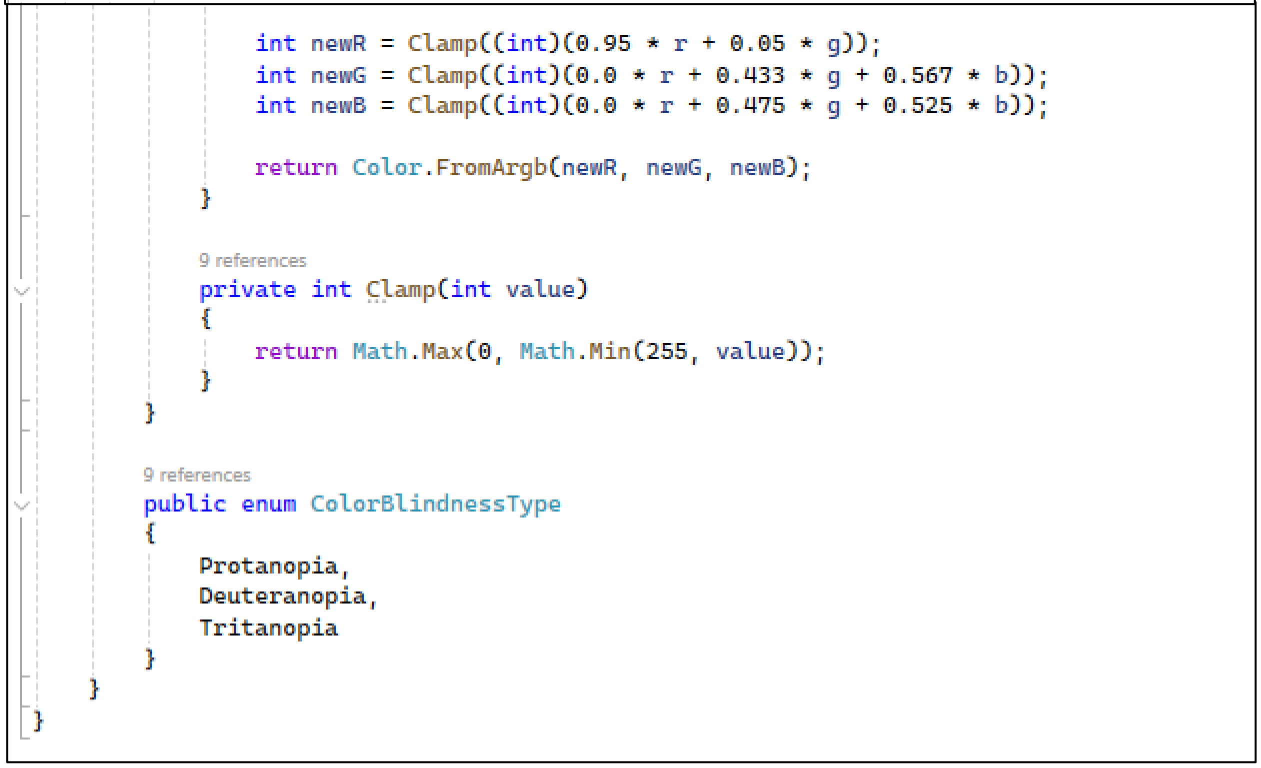
```
1 reference
private Bitmap TransformImage(Bitmap original)
{
    Bitmap transformedImage = new Bitmap(original.Width, original.Height);

    for (int y = 0; y < original.Height; y++)
    {
        for (int x = 0; x < original.Width; x++)
        {
            Color originalColor = original.GetPixel(x, y);
            Color transformedColor = colorTransformer.Transform(originalColor, GetSelectedColorBlindnessType());

            transformedImage.SetPixel(x, y, transformedColor);
        }
    }

    return transformedImage;
}

1 reference
private ColorBlindnessType GetSelectedColorBlindnessType()
{
    if (radioButton1.Checked)
        return ColorBlindnessType.Protanopia;
    else if (radioButton2.Checked)
        return ColorBlindnessType.Deuteranopia;
    else if (radioButton3.Checked)
        return ColorBlindnessType.Tritanopia;
    else
        return ColorBlindnessType.Protanopia; // Default to Protanopia if no radio button is selected
}

// ColorTransformer Class Definition
2 references
public class ColorTransformer
{
    1 reference
    public Color Transform(Color color, ColorBlindnessType type)
    {
        switch (type)
        {
            case ColorBlindnessType.Protanopia:
                return SimulateProtanopia(color);
            case ColorBlindnessType.Deuteranopia:
                return SimulateDeuteranopia(color);
```



```
            case ColorBlindnessType.Tritanopia:
                return SimulateTritanopia(color);
            default:
                return color; // Default behavior, no transformation
        }
    }

    1 reference
    private Color SimulateProtanopia(Color color)
    {
        int r = color.R;
        int g = color.G;
        int b = color.B;

        int newR = Clamp((int)(0.567 * r + 0.433 * g));
        int newG = Clamp((int)(0.558 * r + 0.442 * g));
        int newB = Clamp((int)(0.0 * r + 0.242 * g + 0.758 * b));

        return Color.FromArgb(newR, newG, newB);
    }

    1 reference
    private Color SimulateDeuteranopia(Color color)
    {
        int r = color.R;
        int g = color.G;
        int b = color.B;

        int newR = Clamp((int)(0.625 * r + 0.375 * g));
        int newG = Clamp((int)(0.7 * r + 0.3 * g));
        int newB = Clamp((int)(0.0 * r + 0.3 * g + 0.7 * b));

        return Color.FromArgb(newR, newG, newB);
    }
```



```
        private Color SimulateTritanopia(Color color)
        {
            int r = color.R;
            int g = color.G;
            int b = color.B;
```



```
            int newR = Clamp((int)(0.95 * r + 0.05 * g));
            int newG = Clamp((int)(0.0 * r + 0.433 * g + 0.567 * b));
            int newB = Clamp((int)(0.0 * r + 0.475 * g + 0.525 * b));

            return Color.FromArgb(newR, newG, newB);
        }

        9 references
        private int Clamp(int value)
        {
            return Math.Max(0, Math.Min(255, value));
        }
    }

    9 references
    public enum ColorBlindnessType
    {
        Protanopia,
        Deuteranopia,
        Tritanopia
    }
}
}
```

5.6 COLOR CORRECTOR

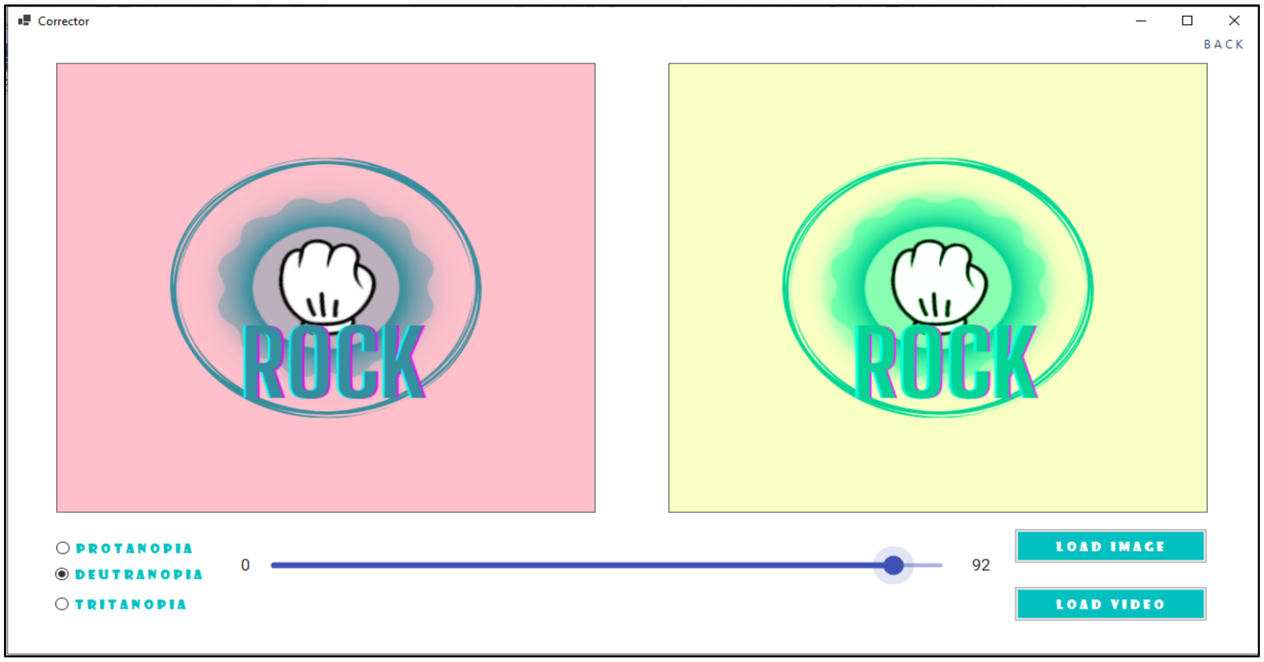
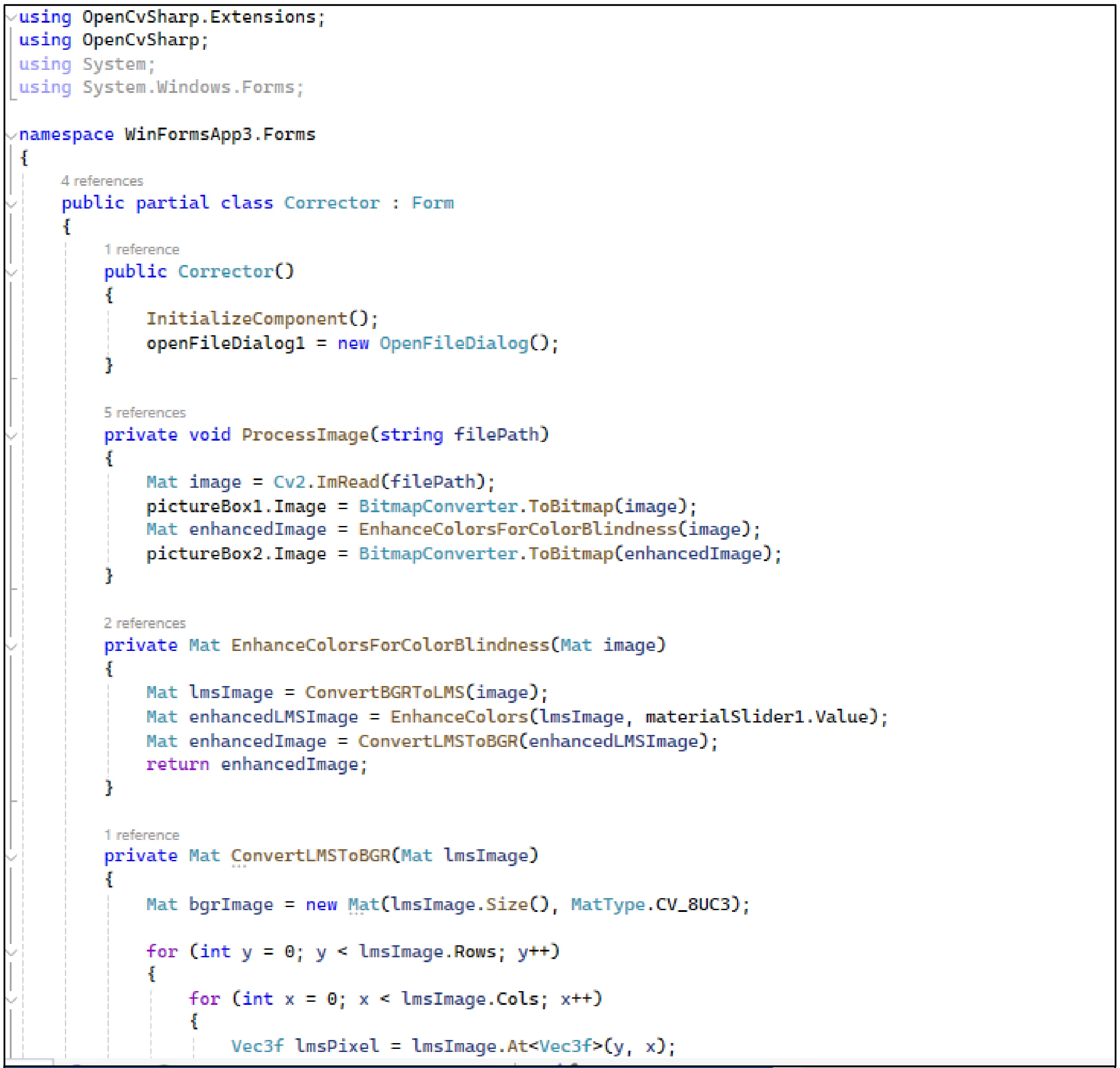
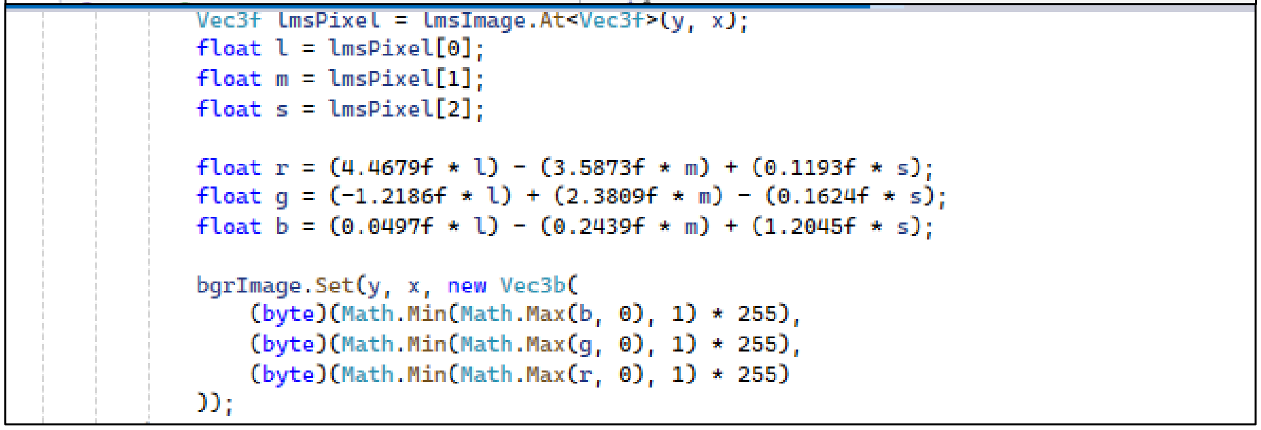


Figure.11

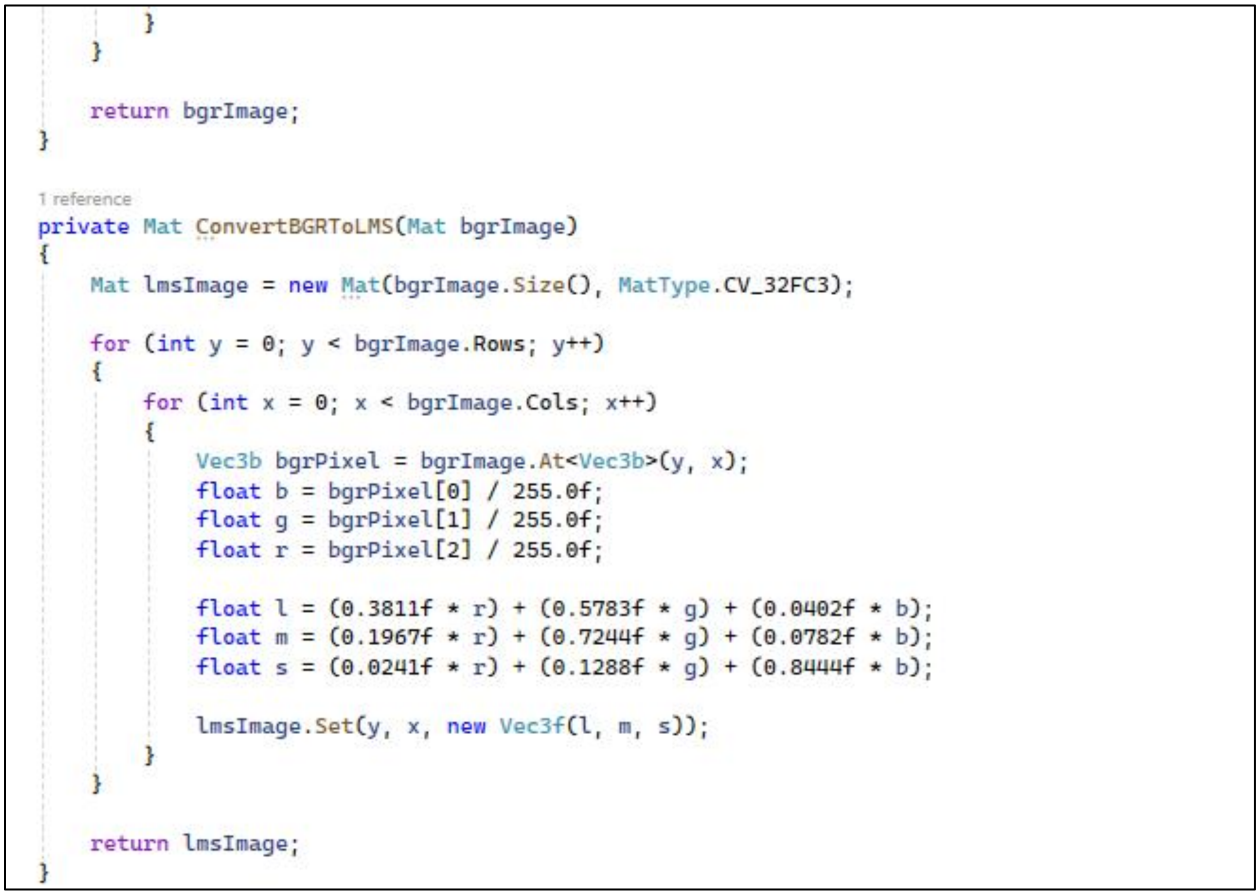
Code:



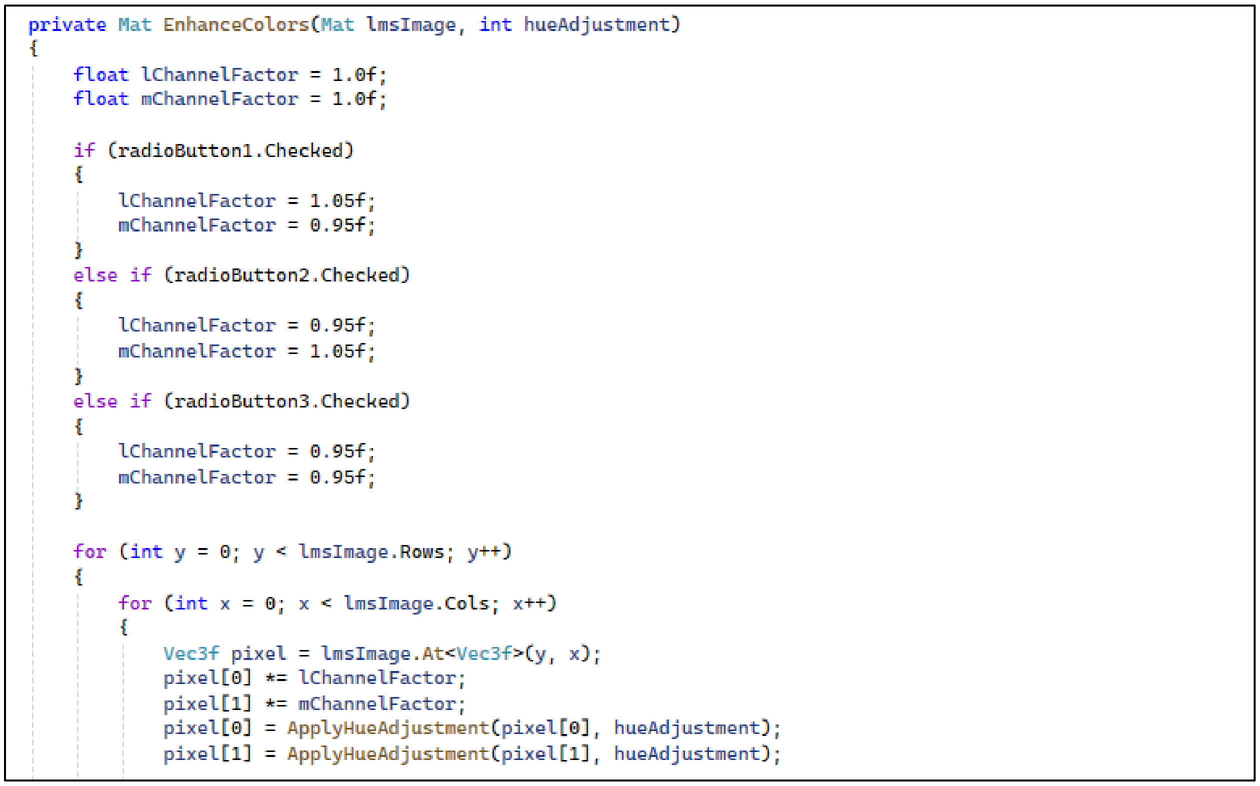
```
using OpenCvSharp.Extensions;
using OpenCvSharp;
using System;
using System.Windows.Forms;

namespace WinFormsApp3.Forms
{
    4 references
    public partial class Corrector : Form
    {
        1 reference
        public Corrector()
        {
            InitializeComponent();
            openFileDialog1 = new OpenFileDialog();
        }

        5 references
        private void ProcessImage(string filePath)
        {
            Mat image = Cv2.ImRead(filePath);
            pictureBox1.Image = BitmapConverter.ToBitmap(image);
            Mat enhancedImage = EnhanceColorsForColorBlindness(image);
            pictureBox2.Image = BitmapConverter.ToBitmap(enhancedImage);
        }

        2 references
        private Mat EnhanceColorsForColorBlindness(Mat image)
        {
            Mat lmsImage = ConvertBGRToLMS(image);
            Mat enhancedLMSImage = EnhanceColors(lmsImage, materialSlider1.Value);
            Mat enhancedImage = ConvertLMSToBGR(enhancedLMSImage);
            return enhancedImage;
        }

        1 reference
        private Mat ConvertLMSToBGR(Mat lmsImage)
        {
            Mat bgrImage = new Mat(lmsImage.Size(), MatType.CV_8UC3);

            for (int y = 0; y < lmsImage.Rows; y++)
            {
                for (int x = 0; x < lmsImage.Cols; x++)
                {
                    Vec3f lmsPixel = lmsImage.At<Vec3f>(y, x);
```



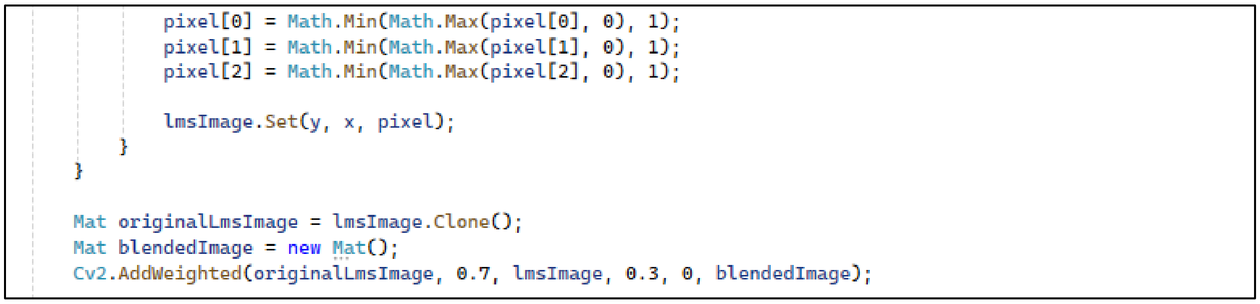
```
                    Vec3f lmsPixel = lmsImage.At<Vec3f>(y, x);
                    float l = lmsPixel[0];
                    float m = lmsPixel[1];
                    float s = lmsPixel[2];

                    float r = (4.4679f * l) - (3.5873f * m) + (0.1193f * s);
                    float g = (-1.2186f * l) + (2.3809f * m) - (0.1624f * s);
                    float b = (0.0497f * l) - (0.2439f * m) + (1.2045f * s);

                    bgrImage.Set(y, x, new Vec3b(
                        (byte)(Math.Min(Math.Max(b, 0), 1) * 255),
                        (byte)(Math.Min(Math.Max(g, 0), 1) * 255),
                        (byte)(Math.Min(Math.Max(r, 0), 1) * 255)
                    ));
```



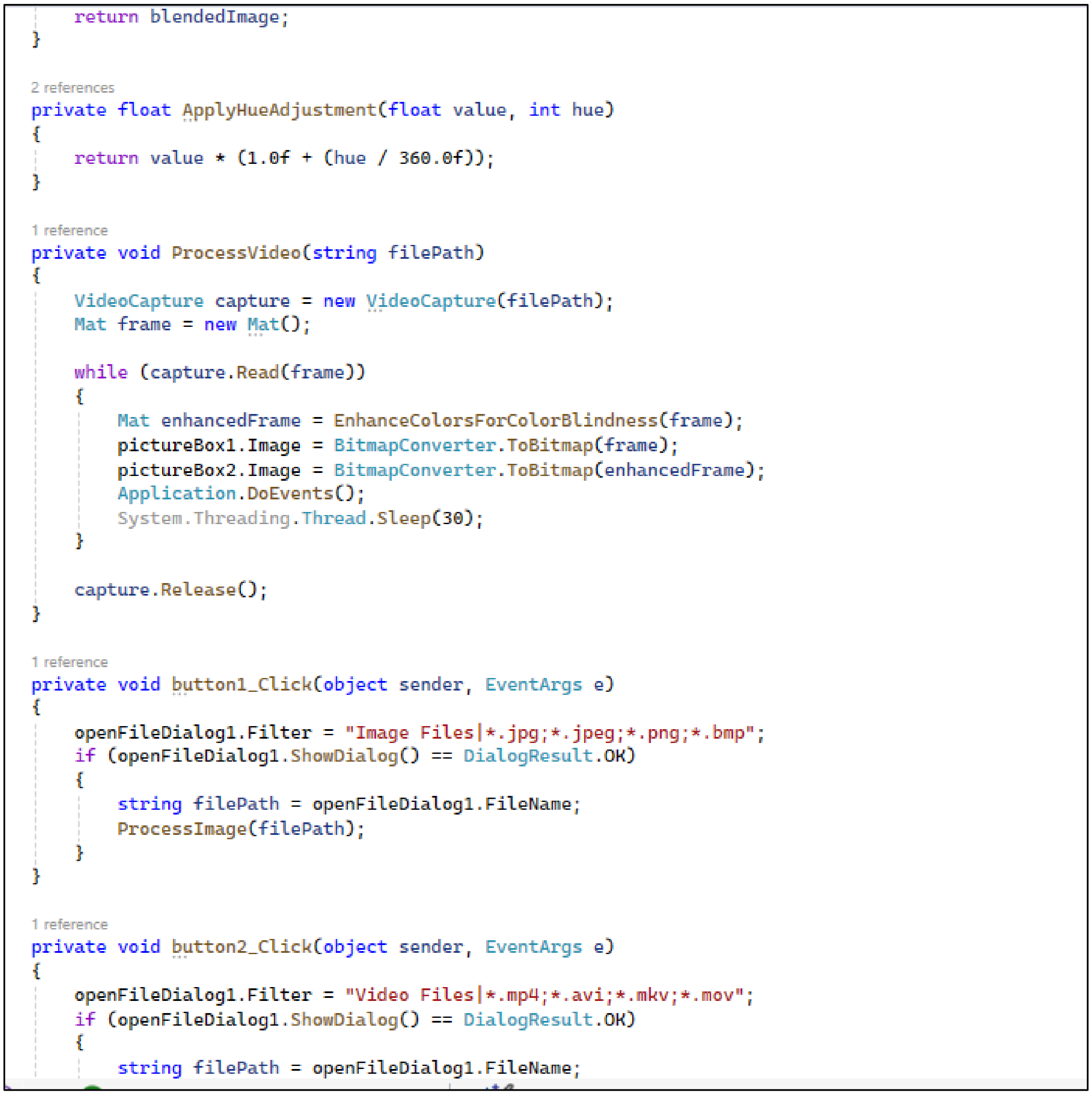
```
            }
        }

        return bgrImage;
    }

    1 reference
    private Mat ConvertBGRToLMS(Mat bgrImage)
    {
        Mat lmsImage = new Mat(bgrImage.Size(), MatType.CV_32FC3);

        for (int y = 0; y < bgrImage.Rows; y++)
        {
            for (int x = 0; x < bgrImage.Cols; x++)
            {
                Vec3b bgrPixel = bgrImage.At<Vec3b>(y, x);
                float b = bgrPixel[0] / 255.0f;
                float g = bgrPixel[1] / 255.0f;
                float r = bgrPixel[2] / 255.0f;

                float l = (0.3811f * r) + (0.5783f * g) + (0.0402f * b);
                float m = (0.1967f * r) + (0.7244f * g) + (0.0782f * b);
                float s = (0.0241f * r) + (0.1288f * g) + (0.8444f * b);

                lmsImage.Set(y, x, new Vec3f(l, m, s));
            }
        }

        return lmsImage;
    }
```



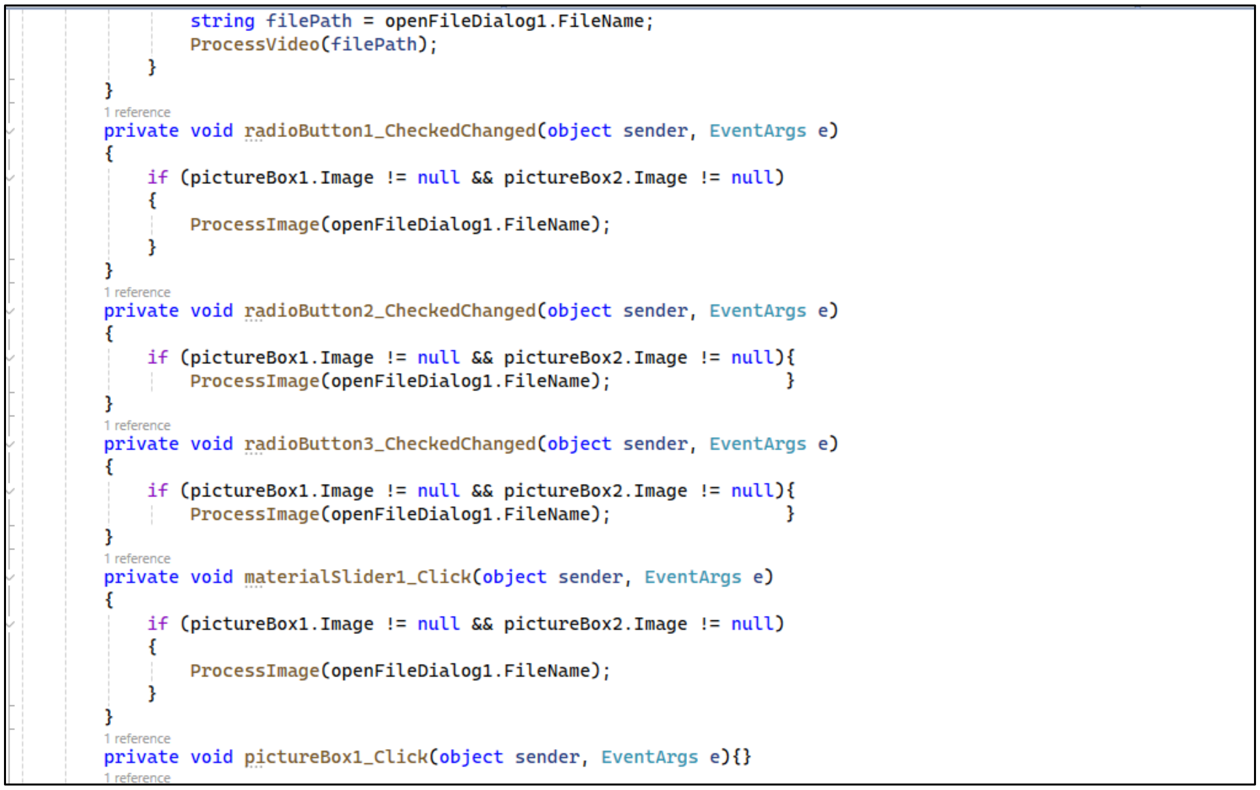
```
private Mat EnhanceColors(Mat lmsImage, int hueAdjustment)
{
    float lChannelFactor = 1.0f;
    float mChannelFactor = 1.0f;

    if (radioButton1.Checked)
    {
        lChannelFactor = 1.05f;
        mChannelFactor = 0.95f;
    }
    else if (radioButton2.Checked)
    {
        lChannelFactor = 0.95f;
        mChannelFactor = 1.05f;
    }
    else if (radioButton3.Checked)
    {
        lChannelFactor = 0.95f;
        mChannelFactor = 0.95f;
    }

    for (int y = 0; y < lmsImage.Rows; y++)
    {
        for (int x = 0; x < lmsImage.Cols; x++)
        {
            Vec3f pixel = lmsImage.At<Vec3f>(y, x);
            pixel[0] *= lChannelFactor;
            pixel[1] *= mChannelFactor;
            pixel[0] = ApplyHueAdjustment(pixel[0], hueAdjustment);
            pixel[1] = ApplyHueAdjustment(pixel[1], hueAdjustment);
```



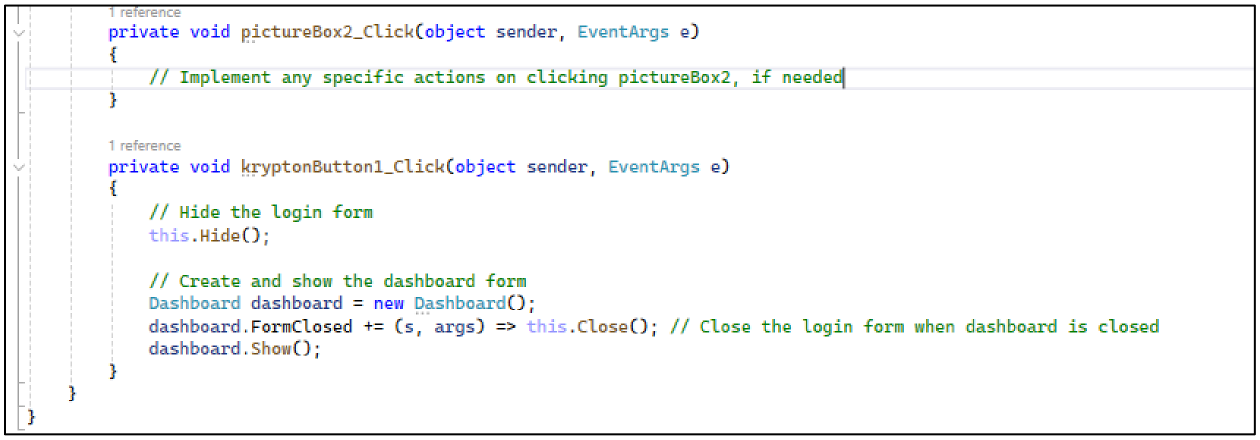
```
                pixel[0] = Math.Min(Math.Max(pixel[0], 0), 1);
                pixel[1] = Math.Min(Math.Max(pixel[1], 0), 1);
                pixel[2] = Math.Min(Math.Max(pixel[2], 0), 1);

                lmsImage.Set(y, x, pixel);
            }
        }

        Mat originalLmsImage = lmsImage.Clone();
        Mat blendedImage = new Mat();
        Cv2.AddWeighted(originalLmsImage, 0.7, lmsImage, 0.3, 0, blendedImage);
```



```
    return blendedImage;
}

2 references
private float ApplyHueAdjustment(float value, int hue)
{
    return value * (1.0f + (hue / 360.0f));
}

1 reference
private void ProcessVideo(string filePath)
{
    VideoCapture capture = new VideoCapture(filePath);
    Mat frame = new Mat();

    while (capture.Read(frame))
    {
        Mat enhancedFrame = EnhanceColorsForColorBlindness(frame);
        pictureBox1.Image = BitmapConverter.ToBitmap(frame);
        pictureBox2.Image = BitmapConverter.ToBitmap(enhancedFrame);
        Application.DoEvents();
        System.Threading.Thread.Sleep(30);
    }

    capture.Release();
}

1 reference
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp";
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog1.FileName;
        ProcessImage(filePath);
    }
}

1 reference
private void button2_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "Video Files|*.mp4;*.avi;*.mkv;*.mov";
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog1.FileName;
```



```
                string filePath = openFileDialog1.FileName;
                ProcessVideo(filePath);
            }
        }
        1 reference
        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            if (pictureBox1.Image != null && pictureBox2.Image != null)
            {
                ProcessImage(openFileDialog1.FileName);
            }
        }
        1 reference
        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            if (pictureBox1.Image != null && pictureBox2.Image != null){
                ProcessImage(openFileDialog1.FileName);                }
        }
        1 reference
        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            if (pictureBox1.Image != null && pictureBox2.Image != null){
                ProcessImage(openFileDialog1.FileName);                }
        }
        1 reference
        private void materialSlider1_Click(object sender, EventArgs e)
        {
            if (pictureBox1.Image != null && pictureBox2.Image != null)
            {
                ProcessImage(openFileDialog1.FileName);
            }
        }
        1 reference
        private void pictureBox1_Click(object sender, EventArgs e){}
        1 reference
```



```
        1 reference
        private void pictureBox2_Click(object sender, EventArgs e)
        {
            // Implement any specific actions on clicking pictureBox2, if needed
        }

        1 reference
        private void kryptonButton1_Click(object sender, EventArgs e)
        {
            // Hide the login form
            this.Hide();

            // Create and show the dashboard form
            Dashboard dashboard = new Dashboard();
            dashboard.FormClosed += (s, args) => this.Close(); // Close the login form when dashboard is closed
            dashboard.Show();
        }
    }
}
```

5.7 COLOR DETECTOR

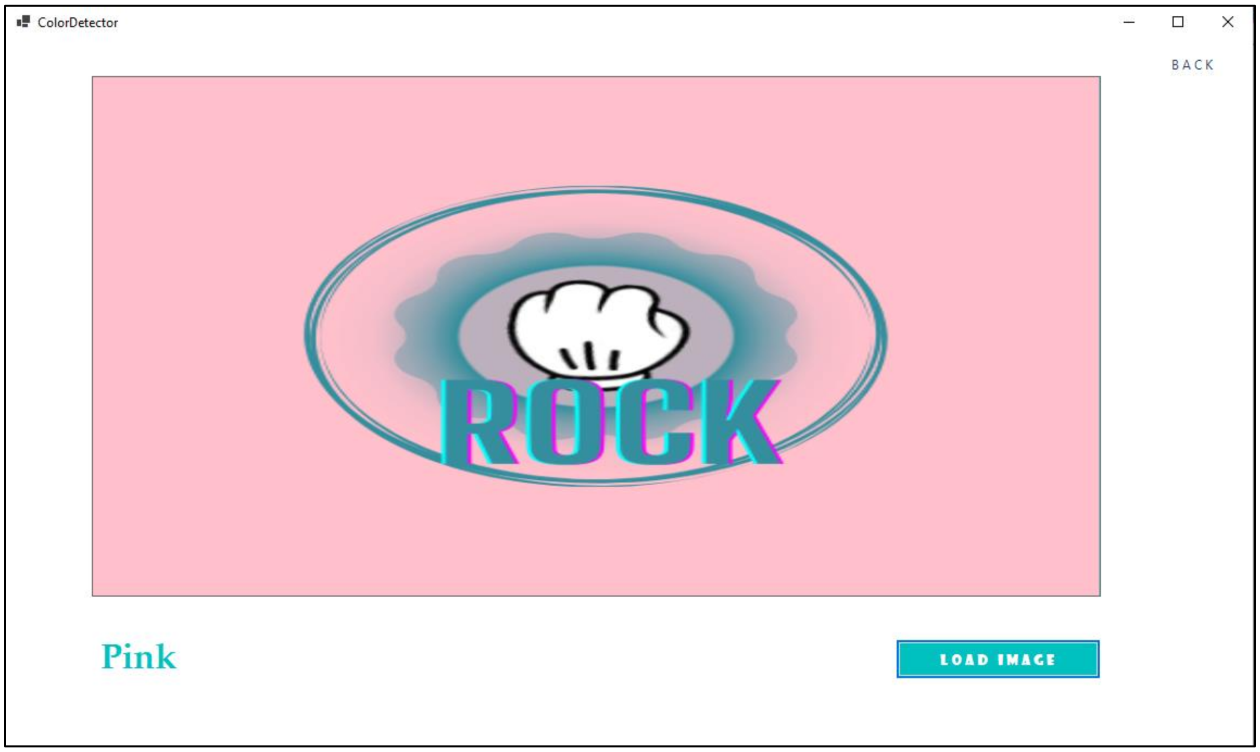
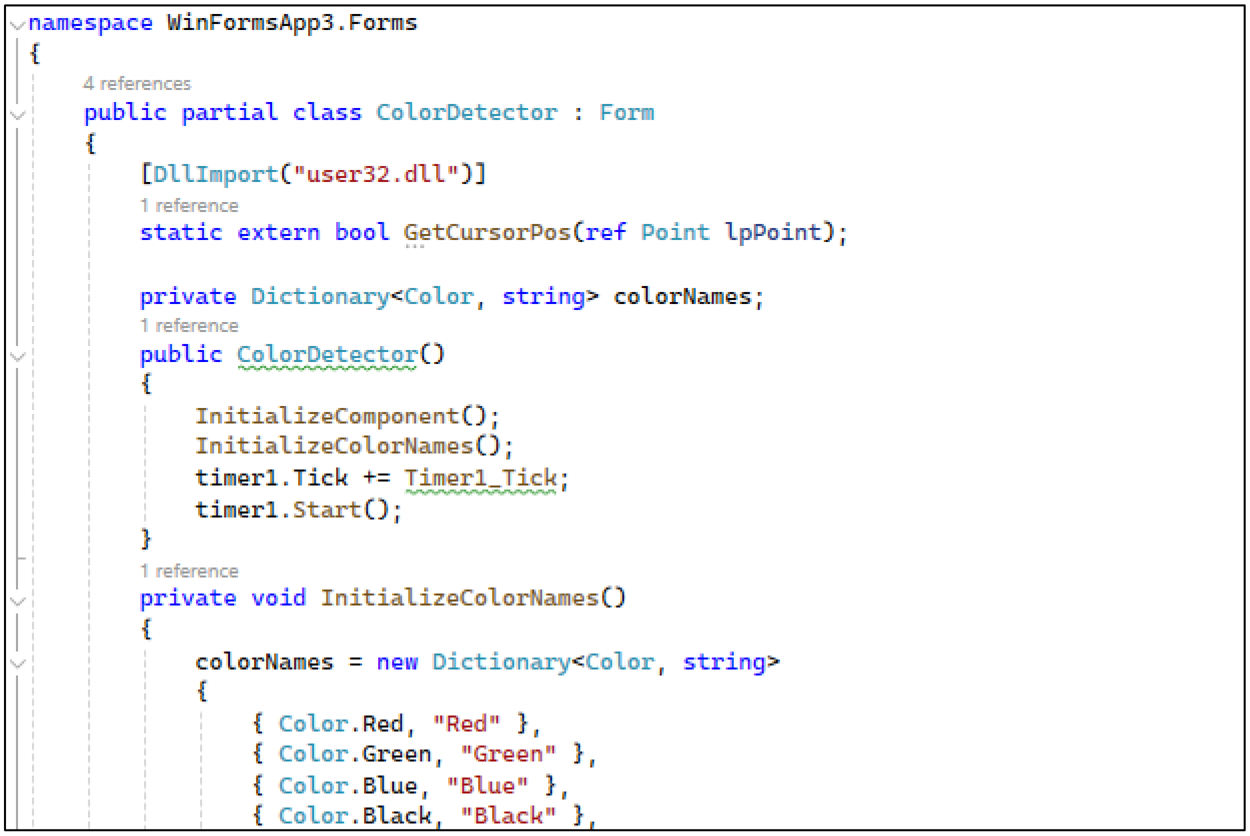
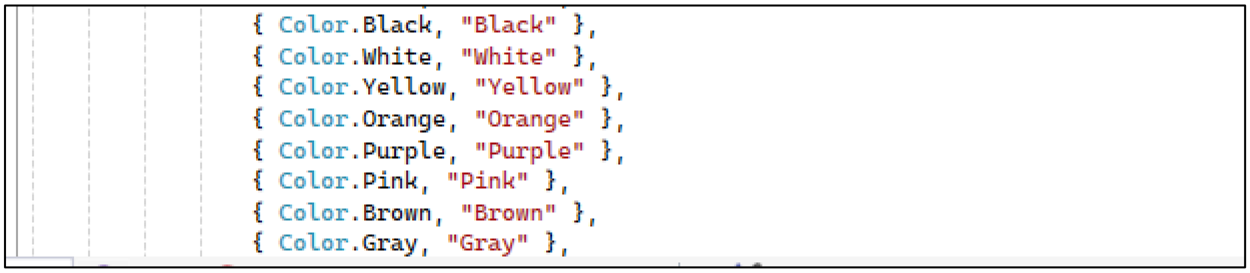


Figure.12

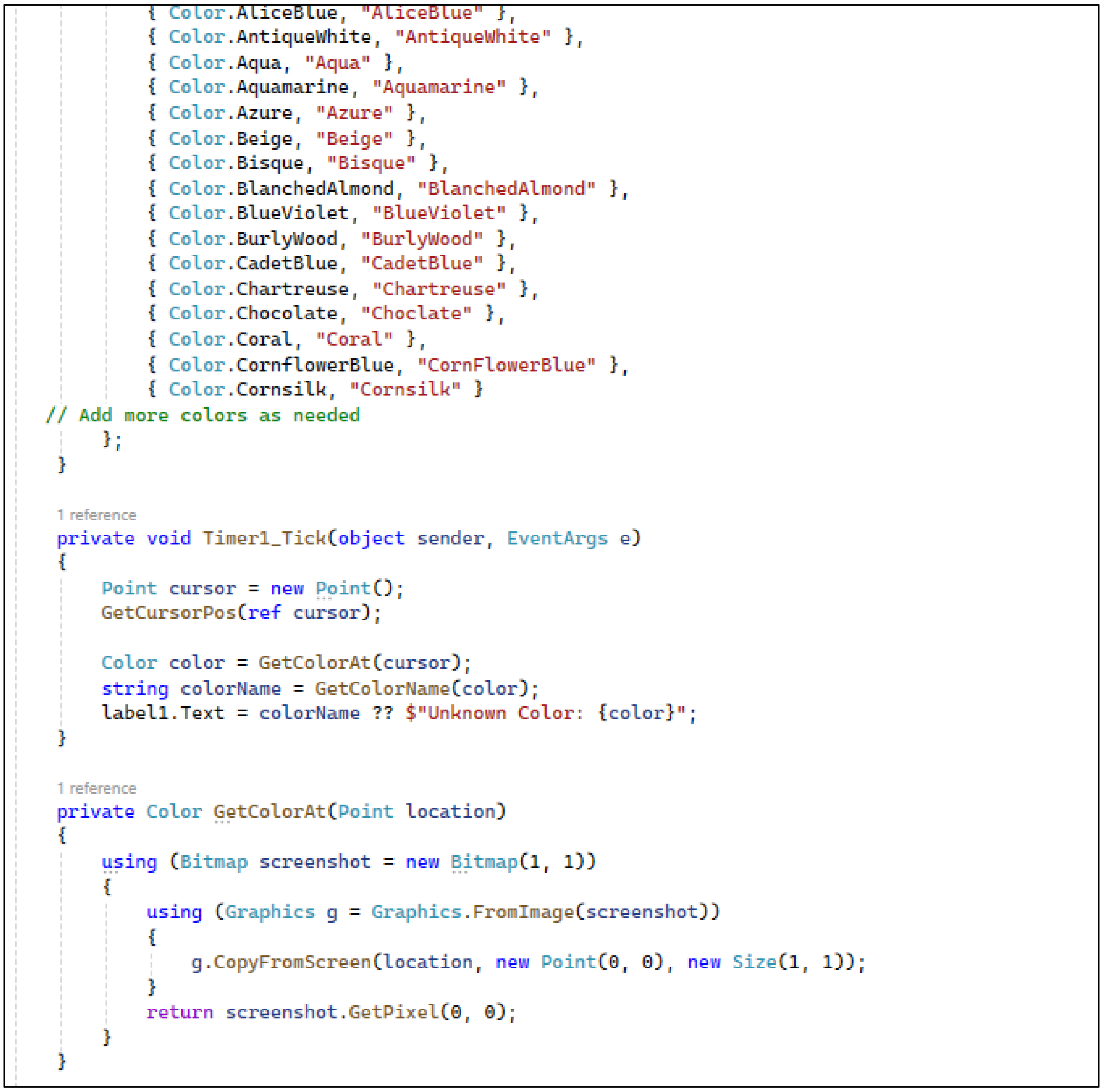
Code:



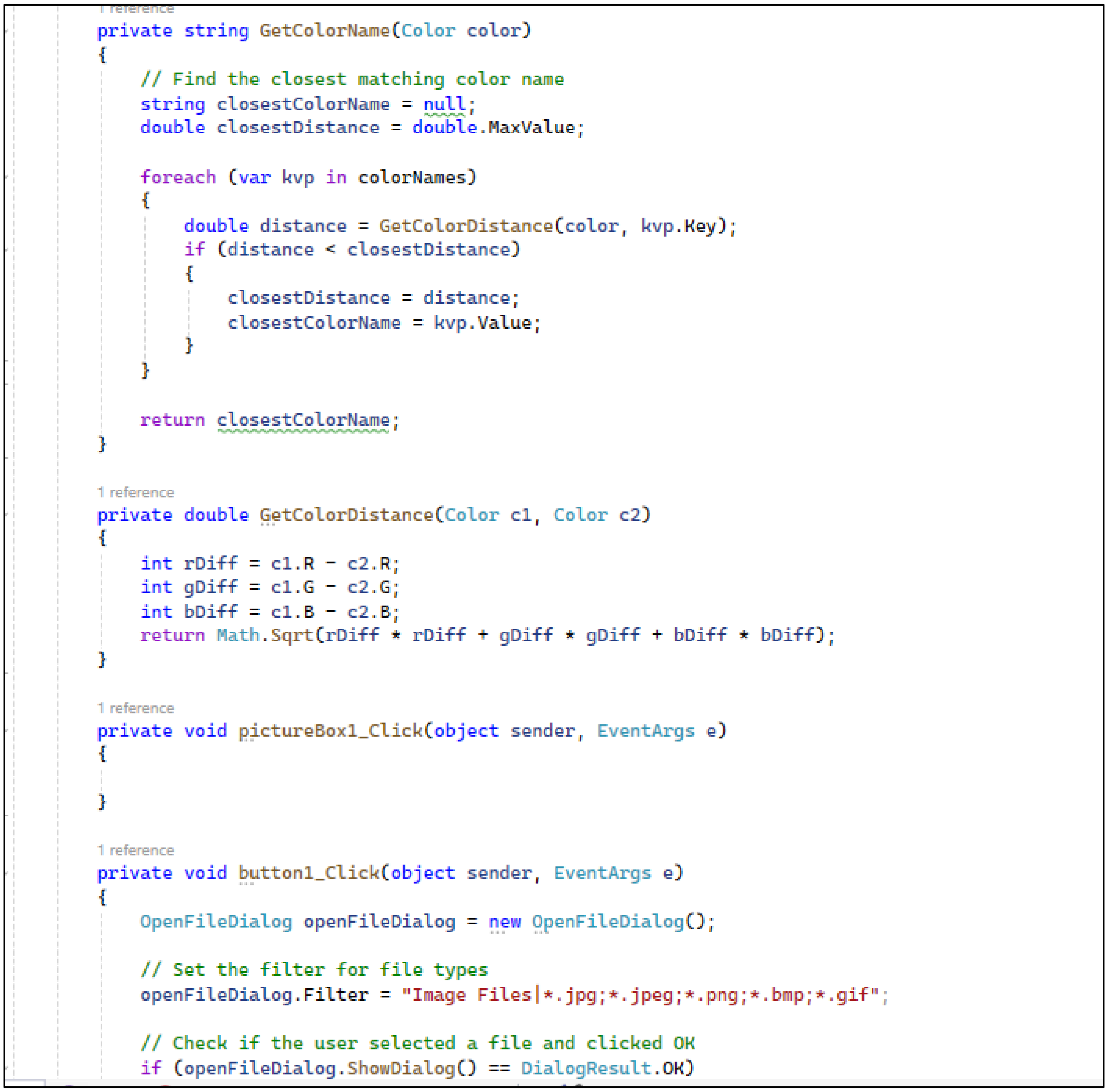
```
namespace WinFormsApp3.Forms
{
    4 references
    public partial class ColorDetector : Form
    {
        [DllImport("user32.dll")]
        1 reference
        static extern bool GetCursorPos(ref Point lpPoint);

        private Dictionary<Color, string> colorNames;
        1 reference
        public ColorDetector()
        {
            InitializeComponent();
            InitializeColorNames();
            timer1.Tick += Timer1_Tick;
            timer1.Start();
        }
        1 reference
        private void InitializeColorNames()
        {
            colorNames = new Dictionary<Color, string>
            {
                { Color.Red, "Red" },
                { Color.Green, "Green" },
                { Color.Blue, "Blue" },
                { Color.Black, "Black" },
```



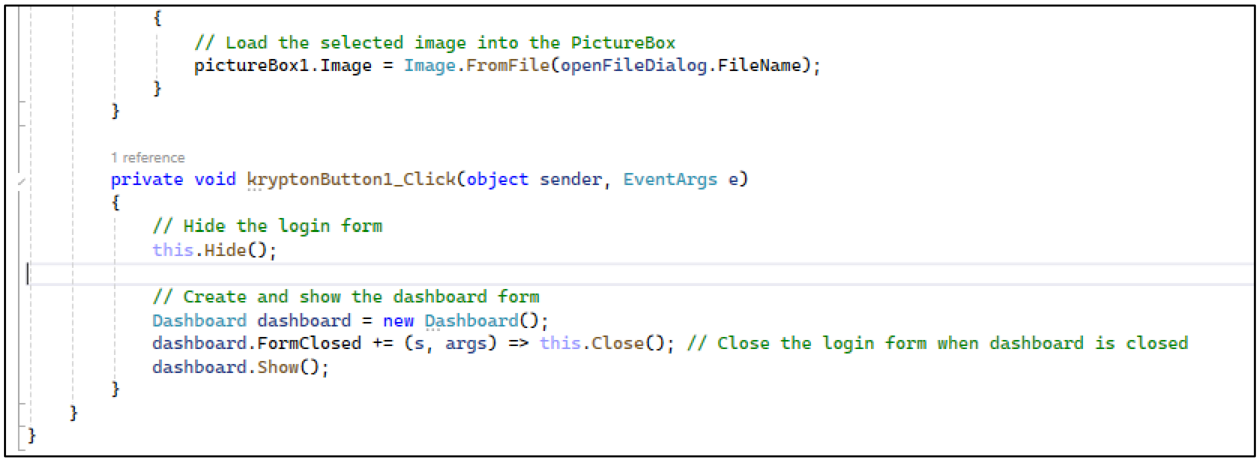
```
            { Color.Black, "Black" },
            { Color.White, "White" },
            { Color.Yellow, "Yellow" },
            { Color.Orange, "Orange" },
            { Color.Purple, "Purple" },
            { Color.Pink, "Pink" },
            { Color.Brown, "Brown" },
            { Color.Gray, "Gray" },
```



```
            { Color.AliceBlue, "AliceBlue" },
            { Color.AntiqueWhite, "AntiqueWhite" },
            { Color.Aqua, "Aqua" },
            { Color.Aquamarine, "Aquamarine" },
            { Color.Azure, "Azure" },
            { Color.Beige, "Beige" },
            { Color.Bisque, "Bisque" },
            { Color.BlanchedAlmond, "BlanchedAlmond" },
            { Color.BlueViolet, "BlueViolet" },
            { Color.BurlyWood, "BurlyWood" },
            { Color.CadetBlue, "CadetBlue" },
            { Color.Chartreuse, "Chartreuse" },
            { Color.Chocolate, "Choclate" },
            { Color.Coral, "Coral" },
            { Color.CornflowerBlue, "CornFlowerBlue" },
            { Color.Cornsilk, "Cornsilk" }
   // Add more colors as needed
        };
    }

    1 reference
    private void Timer1_Tick(object sender, EventArgs e)
    {
        Point cursor = new Point();
        GetCursorPos(ref cursor);

        Color color = GetColorAt(cursor);
        string colorName = GetColorName(color);
        label1.Text = colorName ?? $"Unknown Color: {color}";
    }

    1 reference
    private Color GetColorAt(Point location)
    {
        using (Bitmap screenshot = new Bitmap(1, 1))
        {
            using (Graphics g = Graphics.FromImage(screenshot))
            {
                g.CopyFromScreen(location, new Point(0, 0), new Size(1, 1));
            }
            return screenshot.GetPixel(0, 0);
        }
    }
```

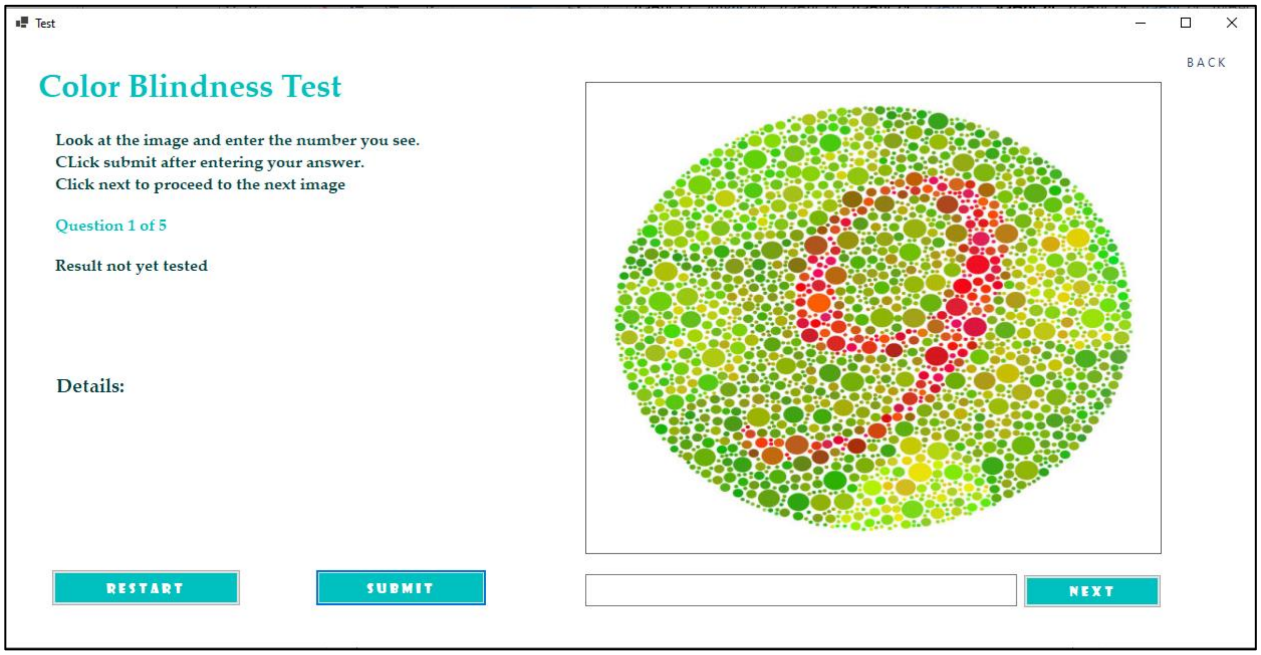


```
1 reference
private string GetColorName(Color color)
{
    // Find the closest matching color name
    string closestColorName = null;
    double closestDistance = double.MaxValue;

    foreach (var kvp in colorNames)
    {
        double distance = GetColorDistance(color, kvp.Key);
        if (distance < closestDistance)
        {
            closestDistance = distance;
            closestColorName = kvp.Value;
        }
    }

    return closestColorName;
}

1 reference
private double GetColorDistance(Color c1, Color c2)
{
    int rDiff = c1.R - c2.R;
    int gDiff = c1.G - c2.G;
    int bDiff = c1.B - c2.B;
    return Math.Sqrt(rDiff * rDiff + gDiff * gDiff + bDiff * bDiff);
}

1 reference
private void pictureBox1_Click(object sender, EventArgs e)
{

}

1 reference
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();

    // Set the filter for file types
    openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp;*.gif";

    // Check if the user selected a file and clicked OK
    if (openFileDialog.ShowDialog() == DialogResult.OK)
```



```
        {
            // Load the selected image into the PictureBox
            pictureBox1.Image = Image.FromFile(openFileDialog.FileName);
        }
    }

    1 reference
    private void kryptonButton1_Click(object sender, EventArgs e)
    {
        // Hide the login form
        this.Hide();

        // Create and show the dashboard form
        Dashboard dashboard = new Dashboard();
        dashboard.FormClosed += (s, args) => this.Close(); // Close the login form when dashboard is closed
        dashboard.Show();
    }
}
}
```

5.8 CVD TESTING



Look at the image and enter the number you see.
CLick submit after entering your answer.
Click next to proceed to the next image

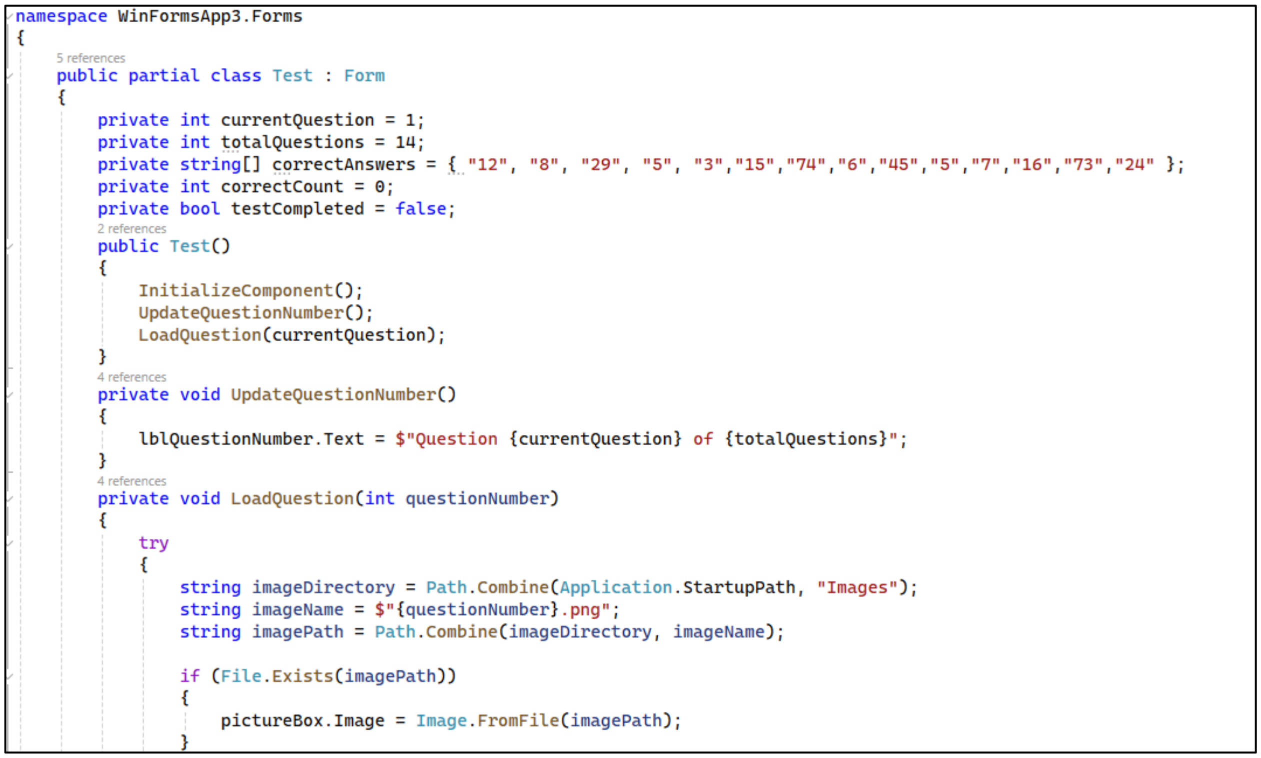
Question 1 of 5

Result not yet tested

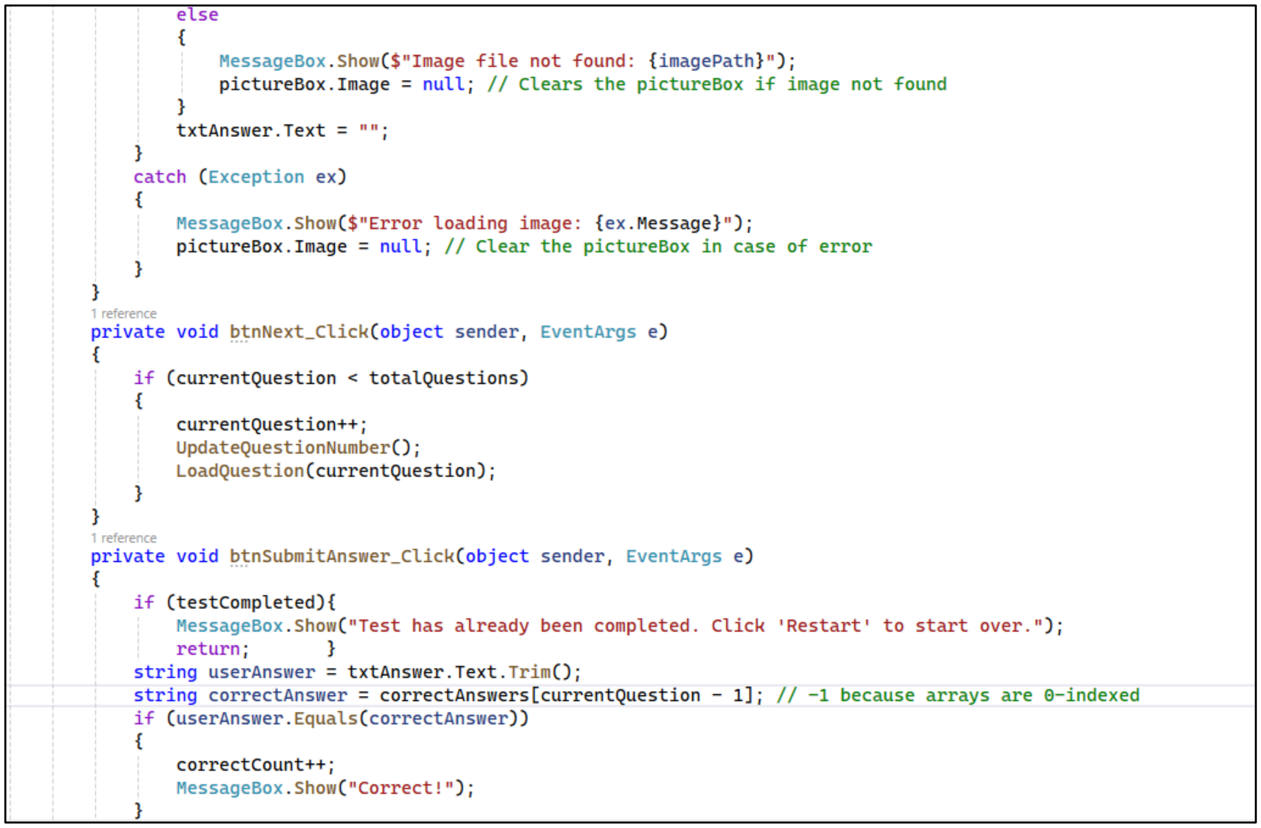
Details:

Figure.12

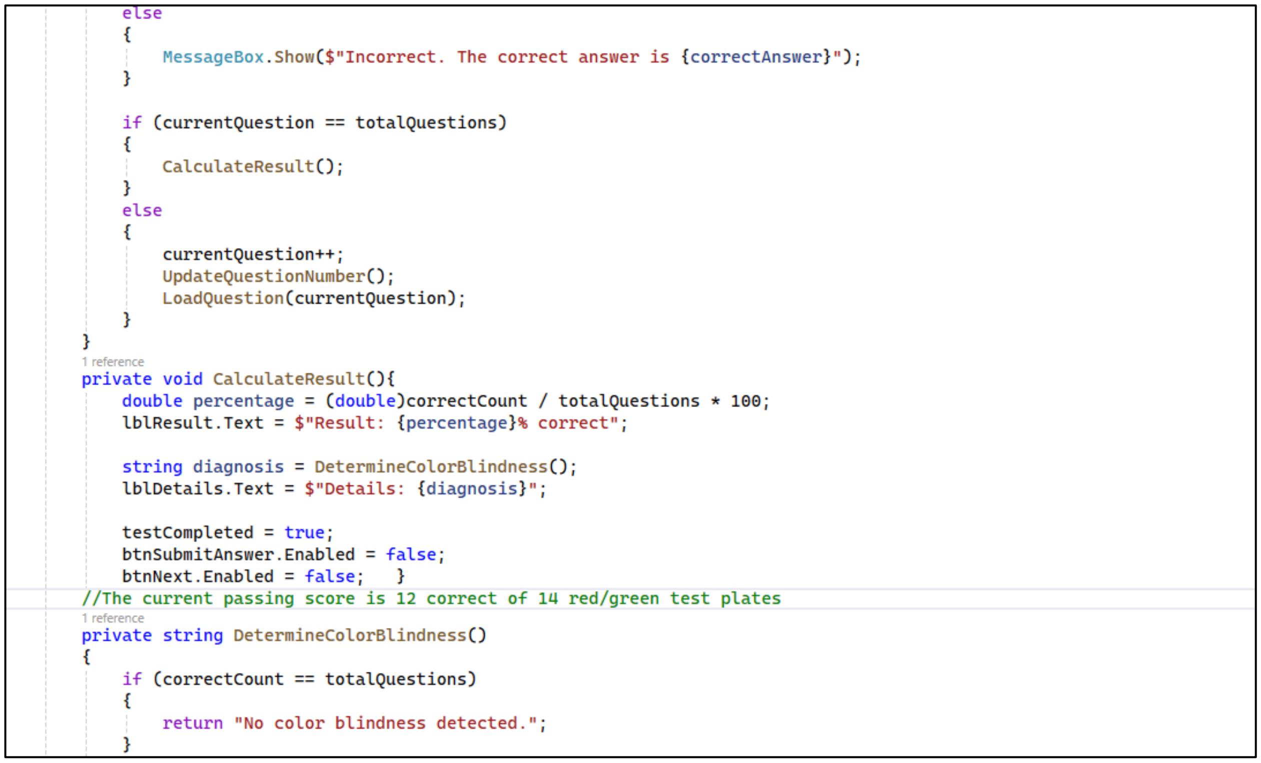
Code:



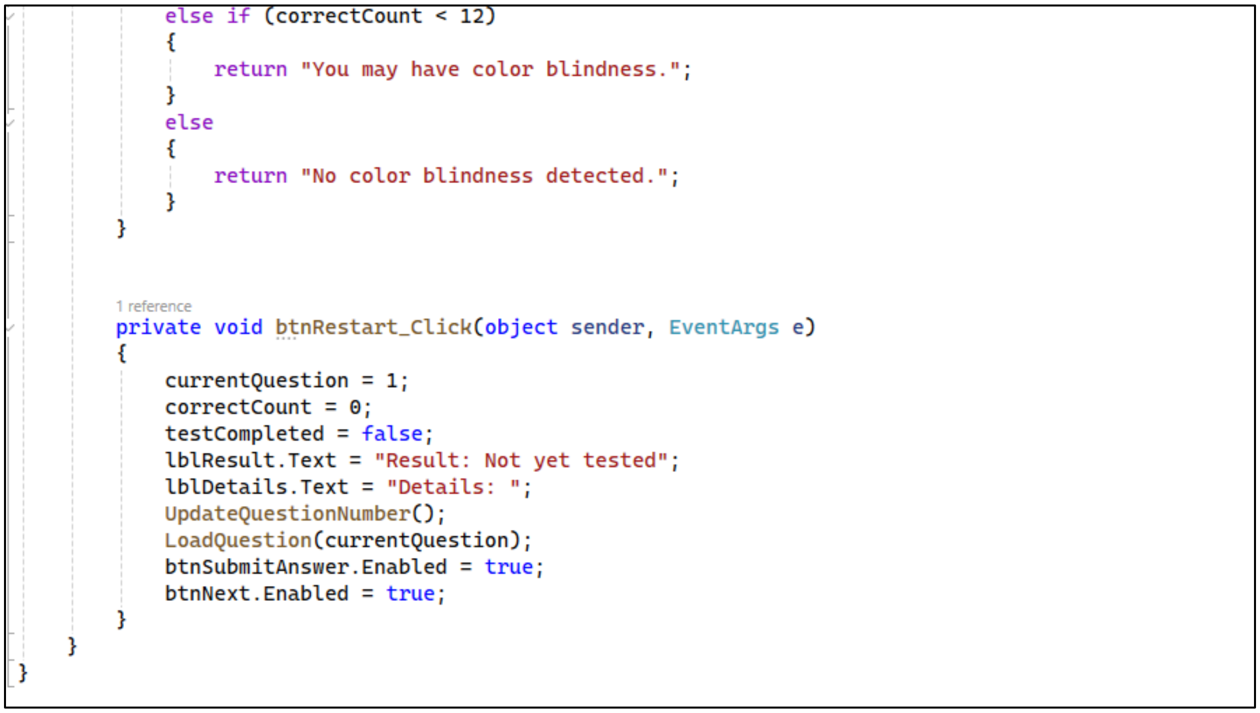
```
namespace WinFormsApp3.Forms
{
    5 references
    public partial class Test : Form
    {
        private int currentQuestion = 1;
        private int totalQuestions = 14;
        private string[] correctAnswers = { "12", "8", "29", "5", "3","15","74","6","45","5","7","16","73","24" };
        private int correctCount = 0;
        private bool testCompleted = false;
        2 references
        public Test()
        {
            InitializeComponent();
            UpdateQuestionNumber();
            LoadQuestion(currentQuestion);
        }
        4 references
        private void UpdateQuestionNumber()
        {
            lblQuestionNumber.Text = $"Question {currentQuestion} of {totalQuestions}";
        }
        4 references
        private void LoadQuestion(int questionNumber)
        {
            try
            {
                string imageDirectory = Path.Combine(Application.StartupPath, "Images");
                string imageName = $"{questionNumber}.png";
                string imagePath = Path.Combine(imageDirectory, imageName);

                if (File.Exists(imagePath))
                {
                    pictureBox.Image = Image.FromFile(imagePath);
                }
```



```
                else
                {
                    MessageBox.Show($"Image file not found: {imagePath}");
                    pictureBox.Image = null; // Clears the pictureBox if image not found
                }
                txtAnswer.Text = "";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error loading image: {ex.Message}");
                pictureBox.Image = null; // Clear the pictureBox in case of error
            }
        }
        1 reference
        private void btnNext_Click(object sender, EventArgs e)
        {
            if (currentQuestion < totalQuestions)
            {
                currentQuestion++;
                UpdateQuestionNumber();
                LoadQuestion(currentQuestion);
            }
        }
        1 reference
        private void btnSubmitAnswer_Click(object sender, EventArgs e)
        {
            if (testCompleted){
                MessageBox.Show("Test has already been completed. Click 'Restart' to start over.");
                return;        }
            string userAnswer = txtAnswer.Text.Trim();
            string correctAnswer = correctAnswers[currentQuestion - 1]; // -1 because arrays are 0-indexed
            if (userAnswer.Equals(correctAnswer))
            {
                correctCount++;
                MessageBox.Show("Correct!");
            }
```



```
            else
            {
                MessageBox.Show($"Incorrect. The correct answer is {correctAnswer}");
            }

            if (currentQuestion == totalQuestions)
            {
                CalculateResult();
            }
            else
            {
                currentQuestion++;
                UpdateQuestionNumber();
                LoadQuestion(currentQuestion);
            }
        }
        1 reference
        private void CalculateResult(){
            double percentage = (double)correctCount / totalQuestions * 100;
            lblResult.Text = $"Result: {percentage}% correct";

            string diagnosis = DetermineColorBlindness();
            lblDetails.Text = $"Details: {diagnosis}";

            testCompleted = true;
            btnSubmitAnswer.Enabled = false;
            btnNext.Enabled = false;   }
        //The current passing score is 12 correct of 14 red/green test plates
        1 reference
        private string DetermineColorBlindness()
        {
            if (correctCount == totalQuestions)
            {
                return "No color blindness detected.";
            }
```



```
                else if (correctCount < 12)
                {
                    return "You may have color blindness.";
                }
                else
                {
                    return "No color blindness detected.";
                }
            }


            1 reference
            private void btnRestart_Click(object sender, EventArgs e)
            {
                currentQuestion = 1;
                correctCount = 0;
                testCompleted = false;
                lblResult.Text = "Result: Not yet tested";
                lblDetails.Text = "Details: ";
                UpdateQuestionNumber();
                LoadQuestion(currentQuestion);
                btnSubmitAnswer.Enabled = true;
                btnNext.Enabled = true;
            }
        }
}
```

6. Challenges faced

- First of all, implementing accurate simulation and correction for different types of color blindness required sophisticated image processing techniques. Fine-tuning the transformation matrices to reflect realistic color perception for various deficiencies was particularly challenging.
- Ensuring that the application performed efficiently in real-time, especially for features like the color label and simulator, was a significant hurdle. Optimizing the code to reduce latency and improve responsiveness was essential.
- Combining the simulator, corrector, color label, and tester into a single cohesive application posed integration challenges. Ensuring seamless interaction between these features while maintaining overall stability required meticulous planning and execution.
- Validating the accuracy and effectiveness of the application was difficult, especially without access to a large pool of color-blind testers. Relying on empirical data and feedback from a limited number of users made it challenging to ensure comprehensive testing.

7. Future Improvements

- Further refinement of the transformation matrices and algorithms can improve the realism and accuracy of the color blindness simulation, making it more beneficial for educational and training purposes.
- Incorporating additional color blindness tests beyond the Ishihara test, such as the Farnsworth-Munsell 100 Hue Test, could provide a more comprehensive assessment of users' color vision capabilities.
- Developing mobile and web-based versions of the application would increase accessibility, allowing users to benefit from the tools on various devices and platforms.
- Exploring integration with other assistive technologies, such as screen readers and voice assistants, can provide a more inclusive experience for users with multiple disabilities.
- Conducting extensive user testing with a diverse group of color-blind individuals will help identify areas for improvement and validate the application's effectiveness in real-world scenarios.

8. Conclusion

Our project, the "Experience the Unseen: Color Clarity Simulation System" represents a significant step towards enhancing the daily lives of individuals with color blindness. By integrating simulation, correction, color identification, and testing into a single application, we have provided a valuable toolset that addresses multiple challenges faced by color-blind users. Our project demonstrates the potential of advanced image processing that foster independence and confidence among individuals with color vision deficiencies.

While there are still areas for improvement and expansion, the current application lays a strong foundation for future development. By continuing to refine the system and incorporating user feedback, we can further enhance its impact and reach. Our project underscores the importance of technological innovation in promoting accessibility and inclusivity, ultimately contributing to a more equitable society for all.

9. REFERENCES

- Color Blindness Simulation
  (URL:https://en.wikipedia.org/wiki/Color_blindness#Simulation)

- Correct or simulate color blindness
  (URL: http://www.daltonize.org/)

- M. P. Simunovic, "Colour vision deficiency," Eye (London), vol. 24, no. 5, pp. 747-755, May 2010.(URL : https://pubmed.ncbi.nlm.nih.gov/19927164/)

- Microsoft Developer Documentation on Color Structures and Operations
  URL : MSDN Color Structure

- Stack Overflow discussions and code snippets on RGB color comparison and Euclidean distance calculations: Stack Overflow RGB Comparison